Certified Tester
Game Testing (CT-GaMe)
Syllabus

Version 2.0

International Software Testing Qualifications Board



Provided by

Russian Software Testing Qualifications Board (RSTQB)



Copyright Notice

Copyright Notice © International Software Testing Qualifications Board (hereinafter called ISTQB®)

ISTQB® is a registered trademark of the International Software Testing Qualifications Board.

Copyright © 2026, the authors Andrey Konushin (chair), Alexander Alexandrov, Alexander Torgovkin, Vadim Lukovaty, Kseniya Kondakova, Jason Miller.

Copyright © 2022, the authors Andrey Konushin (chair), Alexander Alexandrov, Evgeny Glushkin, Elizaveta Kruchinina, Vadim Lukovaty, Dmitry Melishev, Maxim Nikolaev, Alexander Prokhorov, Anton Savvateev, Ayrat Sayfullov, Pavel Sharikov, Kirill Shevelev, Artyom Stukalov, Nikita Sysuev, Tatiana Tepaeva, Alexander Torgovkin, Margarita Trofimova, Yaroslav Vereshchagin, Svetlana Yushina, Lyubov Zhuravleva.

All rights reserved. The authors hereby transfer the copyright to the ISTQB®. The authors (as current copyright holders) and ISTQB® (as the future copyright holder) have agreed to the following conditions of use:

Extracts, for non-commercial use, from this document may be copied if the source is acknowledged. Any Accredited Training Provider may use this syllabus as the basis for a training course if the authors and the ISTQB® are acknowledged as the source and copyright owners of the syllabus and provided that any advertisement of such a training course may mention the syllabus only after official Accreditation of the training materials has been received from an ISTQB®-recognized Member Board.

Any individual or group of individuals may use this syllabus as the basis for articles and books, if the authors and the ISTQB® are acknowledged as the source and copyright owners of the syllabus.

Any other use of this syllabus is prohibited without first obtaining the approval in writing of the ISTQB®.

Any ISTQB®-recognized Member Board may translate this syllabus provided they reproduce the Copyright Notice mentioned above in the translated version of the syllabus.

Revision History

Version	Date	Remarks
v1.0.1	October 21, 2022	GA Released
v.2.0	October 30, 2026	GA Released

© International Software Testing Qualifications Board

Table of Contents

Copyright Notice 2
Revision History 3
Table of Contents 4
Acknowledgements 7
0.1 Purpose of this Syllabus 8
0.2 Video Game Testing Qualification 8
0.3 Career Path for Testers 8
0.4 Business Outcomes 8
0.5 Learning Objectives and Cognitive Level of Knowledge 9
0.6 The Game Testing Certificate Exam 9
0.7 Accreditation 10
0.8 Handling of Standards 10
0.9 Level of Detail 10
0.10 How this Syllabus is Organized 10
1 The Specifics of Video Game Testing – 55 minutes 12
1.1 Video Game Testing Basics 12
1.1.1 The Specifics of Video Game Testing 12
1.1.2 Video Game Product Risks 13
1.1.3 Mitigating Video Game Risks by Testing 14
1.1.4 Game-Specific Failures 14
1.1.5 Specific Objectives of Video Game Testing 15
1.2 Typical Roles in a Video Game Development Team 16
1.3 Test Activities Throughout the Game Software Development Lifecycle 17
2 Testing Video Game Mechanics – 120 minutes 19
2.1 Principles and Concepts of Video Game Mechanics 19
2.1.1 Differences Between Testing Gameplay and Non-Gameplay Mechanics 20
2.1.2 Differences Between Testing Core and Meta Mechanics 21
2.1.3 Differences Between Testing Client-Side, Server-Side, and Client-Server Mechanics 22
2.1.4 Types of Video Game Mechanics Failures and Their Possible Causes 23
2.2 Video Game Mechanics Testing 24
2.2.1 Applying Fundamental Approaches to Video Game Mechanics Testing 24
3 Video Game Level Testing - 120 minutes 26
3.1 Principles and Concepts of Level Design in Video Games 26
3.1.1 The Term "Level" and Its Specifics 26
3.1.2 Types of Video Game Level Failures 27

© International Software Testing Qualifications Board

3.2 Test Types in Video Game Levels Testing 30
3.2.1 Geometry Testing 30
3.2.2 Playtesting 31
3.3 Executing Video Game Level Testing 31
3.3.1 Applying Fundamental Approaches to Video Game Level Testing 31
4 Graphics Testing - 95 minutes 34
4.1 Principles and Concepts of Video Game Graphics 35
4.1.1 Features of Video Game Graphics 35
4.1.2 Types of Failures in Video Game Graphics 36
4.2 Test Types in Video Game Graphics Testing 41
4.2.1 Video Game Graphics Artistic Testing 41
4.2.2 Video Game Graphics Technical Testing 42
4.2.3 Video Game Graphics Playtesting 42
4.3 Executing Video Game Graphics Testing 43
5 Audio Testing - 120 minutes 45
5.1 Principles and Concepts of Audio Content in Video Games 45
5.1.1 Features of Video Game Audio 45
5.1.2 Video Game Audio Types 46
5.1.3 Video Game Audio Techniques 47
5.1.4 Types of Audio Failures 47
5.2 Test Types in Video Game Audio Testing 48
5.2.1 Video Game Content-Audio Testing 48
5.2.2 Video Game Audio Technical Testing 49
5.3 Executing Video Game Audio Testing 49
6 AI and Physics Testing - 180 minutes 51
6.1 Principles and Concepts of Artificial Intelligence (AI) and Physics in Video Games 51
6.1.1 Video Game Artificial Intelligence Features 51
6.1.2 Types of AI Failures in Video Games 52
6.1.3 Video Game Physics Features 53
6.1.4 Types of Game Physics Failures 54
6.2 Executing AI and Video Game Physics Testing 55
6.2.1 Executing AI Testing in Video Games 55
6.2.2 Executing Video Game Physics Testing 56
7 Localization Testing - 145 minutes 58
7.1 Principles and Concepts of Video Game Localization 58
7.1.1 Video Game Localization Essentials 58
7.1.2 Features of Video Game Localization 59

© International Software Testing Qualifications Board

7.1.3 Full and Partial Localization 59
7.1.4 Root Causes of Localization Defects 60
7.1.5 Video Game Localization Failures 61
7.2 Test Types in Localization Testing 63
7.2.1 Linguistic Testing and Adaptation Testing 63
7.2.2 Localization Technical Testing 64
7.3 Executing Localization Testing 65
8 Game Controllers Testing - 60 minutes 67
8.1 Principles and Concepts of Video Game Controllers 67
8.1.1 Types of Video Game Controllers 67
8.1.2 Failures Related to the Performance of Video Game Controllers 68
8.2 Test Types in Game Controller Testing 69
8.2.1 Interoperability Testing 70
8.2.2 Performance Testing 70
8.2.3 Ergonomics Testing 70
8.2.4 Testing Video Game Controllers for Compliance 71
8.2.5 Usability Testing 71
8.2.6 Accessibility Testing 72
9 References 73
9.1 Standards 73
9.2 ISTQB documents 73
9.3 Books 73
9.4 Links (Web/Internet) 74
10 Appendix A – Learning Objectives/Cognitive Level of Knowledge 75
11 Appendix B – Business Outcomes Traceability Matrix with Learning Objectives 77
12 Appendix C – Release Notes 80
13 Appendix D – Video Game Specific Terms ..... **Ошибка! Закладка не определена.**
14 Appendix E – Index 81

© International Software Testing Qualifications Board

Acknowledgements

This document was formally released by the General Assembly of the ISTQB® on 30.10.2026.

It was produced by a core team of the Russian Software Testing Qualifications Board (RSTQB): Andrey Konushin (President of the Board), Alexander Alexandrov (Editor-in-Chief), Alexander Torgovkin, Vadim Lukovaty, Jason Miller.

The core team thanks the review team for their suggestions and input. Russian Software Testing Qualifications Board would like to acknowledge and thank LLC "Bytex" for their contribution to the development of this syllabus.

In addition, the core team would like to acknowledge and thank the leaders of the Working Groups for their early and ongoing guidance: Galit Zucker (General Secretary), Florian Fieber (chair, Testing in Particular Domains Working Group), Richard Seidl (vice-chair, Testing in Particular Domains Working Group), Matthias Hamburg (Glossary Working Group) and Tal Pe'er (chair, Exam Working Group).

The following persons participated in the reviewing, commenting and balloting of this syllabus:

Kseniya Kondakova	Emiola Ibironke	Pavel Sharikov
Ilia Kulakov	Matthias Hamburg	Cristina Sobrero
Murian Song	Klaus Skafte	Kenji Onishi
Reifa Tangon	Vipul Kocher	Satomi Yoshizawa
Shinsuke Matsuki	Albert Laura	Gary Mogyorodi
Andrea Horváth	Lukas Piska	Péter Földházi
Mitko Mitev	Aneta Derková	Michael Stahl
Ray Gillespie	Brian De Luca	Earl Burba
Gergely Agnecz	Ioannis Symeonidis	Judy McKay
Kamila Kozakovičová	Korkor Mensah	Mark Rutz
Niranjan Maharajh	Orosz Dénes	Tal Pe'er

© International Software Testing Qualifications Board

0 Introduction to This Syllabus

0.1 Purpose of this Syllabus

This syllabus forms the basis for the ISTQB® Certified Tester Game Testing. The ISTQB® provides this syllabus as follows:

1. To National Boards, to translate into their local language and to accredit training providers. National Boards may adapt the syllabus to their particular language needs and modify the references to adapt to their local publications.

2. To exam providers, to derive examination questions in their local language adapted to the learning objectives for each syllabus.

3. To training providers, to produce courseware and determine appropriate teaching methods.

4. To certification candidates, to prepare for the exam (as part of a training course or independently).

5. To the international software and systems engineering community, to advance the profession of software and systems testing, and as a basis for books and articles.

The ISTQB® may allow other entities to use this syllabus for other purposes, provided they seek and obtain prior written permission.

0.2 Video Game Testing Qualification

The Certified Tester Game Testing qualification is aimed at anyone involved in software testing who wishes to broaden their knowledge of game testing or anyone who wishes to start a specialist career in game testing. The qualification is also aimed at anyone involved in game software engineering who wishes to gain a better understanding of game testing.

The syllabus considers the following principal aspects of game testing:

- Technical aspects,
- Methodological aspects,
- Organizational aspects.

0.3 Career Path for Testers

Building on the Foundation Level, the Video Game Testing qualification supports the definition of career paths for professional testers. A person with the Game Testing certificate will have extended the broad understanding of testing acquired at the Foundation Level to enable him or her to work effectively as a professional tester in a game project.

Those possessing a Game Testing certificate may use the acronym CT-GaMe.

Please visit [ISTQB-Web] for the latest overview of ISTQB®'s career path.

0.4 Business Outcomes

This section lists the Business Outcomes expected of a candidate who has achieved the Game Testing certification.

A Game Testing Certified Tester can…

© International Software Testing Qualifications Board

GaMe-BO1	Describe basic concepts of video game testing
GaMe-BO2	Determine video game risks, goals and requirements under the needs and expectations of stakeholders
GaMe-BO3	Conceptually design, implement and execute basic video game tests
GaMe-BO4	Know the approaches to video game testing and their purpose
GaMe-BO5	Recognize the tools supporting video game testing
GaMe-BO6	Determine how test activities align with the software development lifecycle and reduce the cost of developing and publishing video games

0.5 Learning Objectives and Cognitive Level of Knowledge

Learning Objectives support the business outcomes and are used to create the Game Testing exams.

In general, all contents of this syllabus are examinable, except for the Introduction and Appendices. The exam questions will confirm knowledge of keywords at K1 level (see below) or learning objectives at the respective level of knowledge.

The specific learning objectives and their levels of knowledge are shown at the beginning of each chapter, and classified as follows:

- K1: Remember
- K2: Understand
- K3: Apply

Further details and examples of learning objectives are given in Appendix A.

For all terms listed as keywords just below chapter headings, the correct name and definition from the ISTQB® glossary shall be remembered (K1), even if not explicitly mentioned in the learning objective.

0.6 The Game Testing Certificate Exam

The Game Testing Certificate exam will be based on this syllabus. Answers to exam questions may require the use of material based on more than one section of this syllabus. Standards and books are included as references, but their content is not examinable, beyond what is summarized in the syllabus itself from such standards and books.

Entry Requirement Note: The ISTQB® Foundation Level certificate shall be obtained before taking the ISTQB® Certified Tester Game Testing certification exam.

The entry criterion for taking the ISTQB® Certified Tester Game Testing exam is that candidates are interested in software testing. However, it is strongly recommended that candidates also:

- have at least a minimal background in either game software development or testing, such as 1 year experience as a game software tester.
- take a course that has been accredited to ISTQB® standards (by one of the ISTQB-recognized member boards).

Refer to [ISTQB_EXAM_S&R] Exam Structures and Rules for the Game Testing v2.0 document for further details.

© International Software Testing Qualifications Board

0.7 Accreditation

An ISTQB® Member Board may accredit training providers whose course material follows this syllabus. Training providers should obtain accreditation guidelines from the Member Board or body that performs the accreditation. An accredited course is recognized as conforming to this syllabus and is allowed to have an ISTQB® exam as part of the course.

The accreditation guidelines for this syllabus follow the general Accreditation Guidelines published by the Processes Management and Compliance Working Group.

0.8 Handling of Standards

International standardization organizations like IEEE and ISO have issued standards associated with quality characteristics and software testing. Such standards are referenced in this syllabus. The purpose of these references is to provide a framework (as in the references to ISO 25010 regarding quality characteristics) or to provide a source of additional information if desired by the reader. Please note that ISTQB® syllabi are using the standard documents as reference. Standards documents are not intended for examination. Refer to section 9.

0.9 Level of Detail

The level of detail in this syllabus allows internationally consistent courses and exams. To achieve this goal, the syllabus consists of:

- General instructional objectives describing the intention of the Certified Tester Game Testing Level
- A list of terms that students must be able to recall
- Learning objectives for each knowledge area, describing the cognitive learning outcome to be achieved
- A description of the key concepts, including references to sources such as accepted literature or standards

The syllabus content is not a description of the entire knowledge area of game testing; it reflects the level of detail to be covered in Certified Tester Game Testing training courses. It focuses on test areas and techniques that can apply to most game projects.

The syllabus uses the terminology (i.e. the name and meaning) of the terms used in software testing and quality assurance according to the ISTQB® Glossary.

0.10 How this Syllabus is Organized

The Certified Tester Game Testing syllabus contains eight chapters covering the knowledge necessary to be a Game Testing Specialist.

The top-level heading for each chapter specifies the minimum time for the chapter; timing is not provided below chapter level. For accredited training courses, the syllabus requires a minimum of 895 minutes (14 hours and 55 minutes) of instruction, distributed across the eight chapters as follows:

- Chapter 1: 55 minutes The Specifics of Video Game Testing
  - The tester learns the basic principles related to video game testing, the reasons why video game testing is required, what the test objectives are, and the difference between video game testing and playing the video game.
  - The tester learns about video game sub-systems.
  - The tester understands the game test process, the major activities, and work products.

© International Software Testing Qualifications Board

- Chapter 2: 120 minutes Testing Video Game Mechanics
  - The tester learns to distinguish between different video game mechanics and its test approaches.
  - The tester learns both dynamic and static test techniques used for video game mechanics testing.
- Chapter 3: 120 minutes Video Game Level Testing
  - The tester learns the dependency of the level design and test types and its tool support.
  - The tester learns about video game level testing failures and their root causes.
  - The tester learns about test types in video game level testing.
- Chapter 4: 95 minutes Graphics Testing
  - The tester learns about features and types of graphic content of the video game.
  - The tester learns about video game level testing failures and their root causes.
  - The tester learns about test types in video game graphics testing.
- Chapter 5: 120 minutes Audio Testing
  - The tester learns about types of the audio content of video game.
  - The tester learns about video game level testing failures and their root causes.
  - The tester learns about test types in video game audio testing.
- Chapter 6: 180 minutes AI and Physics Testing
  - The tester learns about types of AI and physics of video game.
  - The tester learns about video game level testing failures and their root causes.
  - The tester learns about test types in video game AI and video game physics testing.
- Chapter 7: 145 minutes Localization Testing
  - The tester learns the basics of localization and internationalization, including their differences and risks.
  - The tester learns about video game level testing failures and their root causes.
  - The tester learns about test types in video game localization testing.
- Chapter 8: 60 minutes Game Controllers Testing
  - The tester learns about types of game controllers including related failures and their possible root causes.

It is important to note that the terms game, computer game, and video game are used synonymously within this syllabus. This verifies readability and consistency throughout the training program. A game is considered strictly as software and should not be confused with sports or gambling activities.

It is advisable to those readers who have a little or no prior experience in video game testing to refer to the ISTQB Glossary [ISTQB_GLOSSARY] before reading further.

© International Software Testing Qualifications Board

1 The Specifics of Video Game Testing – 55 minutes

Keywords

Compatibility, defect, failure, localization, risk

Video Game Specific Keywords

Concept phase, gameplay, live operations, narrative, post-production phase, pre-production phase, production phase, video game

Learning Objectives for Chapter 1

1.1 Video Game Testing Basics

GaMe-1.1.1 (K1) Recognize specifics of video game testing

GaMe-1.1.2 (K1) Recall product risks in video game software

GaMe-1.1.3 (K1) Recall how video game risks can be mitigated by testing.

GaMe-1.1.4 (K2) Give examples of specific failures related to video game testing

GaMe-1.1.5 (K2) Explain the specific test objectives of video game testing

1.2 Typical Roles in a Video Game Development Team

GaMe-1.2.1 (K1) Recognize specific roles and tasks in the video game development team

1.3 Test Activities throughout the Game Software Development Lifecycle

GaMe-1.3.1 (K1) Recall test activities throughout the video game software development lifecycle

1.1 Video Game Testing Basics

1.1.1 The Specifics of Video Game Testing

As a software product, a video game has several distinguishing characteristics compared to other types of software. Unlike many standalone applications, a video game is often implemented as multiple interacting software applications and services that operate together to deliver the complete game experience. From a testing perspective, this means that game behavior may depend on interactions between client-side applications, server-side components, and supporting services, rather than a single executable system.

These characteristics stem from the presence of specific subsystems, which include:

- Graphics subsystem
- Audio subsystem
- Game logic and mechanics subsystem
- Game physics subsystem
- Artificial intelligence (AI) subsystem

Additionally, video games possess unique attributes that define them as a distinct class of software, such as:

- Providing fairness, preventing any group or individual from gaining an unfair advantage through software exploitation.
- Compatibility with various input devices (controllers) to enable intuitive gameplay interaction.

© International Software Testing Qualifications Board

- Multi-platform and cross-platform compatibility.
- Adherence to established stylistic rules, internal lore, or real-world prototypes (where applicable).
- Complex localization processes that go beyond simple language translation.

Due to the interconnected nature of these subsystems and features, defects that emerge in video games are often specific to this software category.

Another fundamental aspect of game testing is the need to validate not only functional suitability but also the gameplay. Understanding deviations in expected game behavior requires testers to have deep domain knowledge across multiple disciplines. A game tester must assess balance, difficulty, narrative consistency, and the interaction of game components. Thus, this role demands not only technical skills but also creative and analytical thinking.

1.1.2 Video Game Product Risks

The rapid growth of the gaming industry has led to the expansion of games across multiple platforms. At the same time, player expectations have risen, making quality a critical success factor.

Since video games are developed within structured software projects, they share common product risks with other types of software. However, due to their unique characteristics, video games are subject to additional specific risks, such as:

- Product complexity
- Need to prevent cheating and unfair advantages
- Market success dependency on subjective player opinions
- Challenges in cross-platform development and controller compatibility
- Uncertainty in predicting the effectiveness of video game mechanics

Product Complexity

All software systems are inherently complex, but video games introduce a unique challenge. Each game subsystem relies on specialized hardware capable of supporting advanced rendering, physics simulations, and real-time computations. These systems must work together seamlessly to deliver an engaging player experience.

Cheating and Exploits

Cheating represents a major product risk that can damage a game's reputation and commercial success. Unfair advantages gained by exploiting vulnerabilities can lead to player frustration, decreased retention rates, and negative community feedback. If cheating is widespread, it can erode trust in the game and deter new players from joining.

Subjectivity of Player Perception

Unlike business applications, where functional suitability is the primary factor for success, game quality is largely determined by user engagement and entertainment value. Players may tolerate minor functional defects but they will abandon a game if it feels boring, repetitive, or visually outdated.

Many purchase decisions are influenced by game reviews, critics, and community sentiment, making negative public perception a significant risk.

Cross-Platform Challenges

Multi-platform compatibility is a common practice, but providing consistent behavior and user experience across different hardware, operating systems, and input methods presents a significant

© International Software Testing Qualifications Board

product risk. Differences in platform capabilities, performance characteristics, and control schemes can lead to platform-specific defects or inconsistent gameplay behavior. [URL1], [URL2]

Modern gaming platforms also support specialized controllers (e.g., racing wheels, VR motion controllers), each requiring unique handling in input processing and user experience design. As a result, testers must consider the increased risk of platform and device specific failures, and apply appropriate test selection and prioritization to address the most critical usage scenarios.

Video Game Mechanics Effectiveness

Video game mechanics define rules and interactions within a game. Predicting how players will perceive and engage with specific mechanics is difficult, making the risk level high for this area.

1.1.3 Mitigating Video Game Risks by Testing

Most modern video games, particularly open-world and live-service games, are too complex to be fully tested under all possible object interactions, events, and variables. As a result, different categories of risk must be mitigated using different test activities.

Analytical and tool-supported test activities help mitigate risks related to large volumes of data, frequent updates, and regression across interconnected systems. Experience-based test activities, such as playtesting and usability testing, help mitigate risks related to subjective player perception, balance, and overall gameplay experience. Non-functional test activities address risks related to performance efficiency, modifiability, and compatibility across platforms and environments.

Risk-based testing supports this approach by helping testers prioritize which risks require deeper or more frequent testing based on their risk likelihood and potential risk impact.

For each condition, two key questions determine priority:

1. How likely is a player to encounter this defect?
2. If encountered, what is the impact on the player's experience and business?

For example, in linear progression games, defects occurring early in the game may be prioritized over those appearing after 50+ hours of playtime, as early-phase defects can drive away new players.

Testers focus on defect clusters rather than isolated defects, allowing developers to resolve systemic defects at the code or architectural level to enhance overall quality. [Nystrom14]

Player onboarding and tutorial phases are especially critical, as first impressions determine long-term retention.

Another crucial factor in mitigating risk is early involvement of the test team in the development process. When testers participate in the design and prototyping phases, they can identify potential risk areas before code is even written. This proactive approach helps shape features with testability in mind, reduces costly rework, and verifies that quality considerations are embedded into the game's foundation — not retrofitted at the end.

1.1.4 Game-Specific Failures

Most enterprise or business software is designed with utilitarian objectives, such as processing data, managing transactions, or supporting organizational workflows. When such systems — for example, accounting applications or online tax platforms — encounter defects or experience outages, users may face inconvenience and organizations may suffer financial or regulatory consequences. However, the overall stakeholder response is typically procedural and rooted in risk management.

© International Software Testing Qualifications Board

In contrast, video game software serves an entertainment function and often carries a strong emotional component. Games represent immersive experiences that deliver escapism, interactivity, and narrative engagement. As a result, failures in game software — particularly those released after extensive public promotion — can provoke strong emotional reactions. Users may not only express dissatisfaction as consumers but also react with personal disappointment, reflecting the perceived breach of trust and expectations established by pre-release marketing and community engagement.

Given the unique complexity of video games, certain failure types are specific to this software category, including:

- 3D model and animation failures, including physics-based interactions.
- Gameplay mechanics failures and unintended interactions.
- Level design defects, such as incorrect collisions or unreachable areas.
- Audio synchronization and sound failures.
- Game world physics inconsistencies (e.g., unrealistic gravity).
- AI behavior failures, affecting non-player characters (NPCs).
- Narrative or quest logic failures, leading to broken story progression.
- Cultural, legal, and ethical localization failures.
- Controller and input device compatibility failures.

Since games are interactive experiences, failures directly impact immersion and player satisfaction. Unlike productivity software, where usability and efficiency are paramount, game testing focuses on maintaining engaging and fair gameplay.

1.1.5 Specific Objectives of Video Game Testing

Video game testing differs from any other software testing due to the unique characteristics of video games as interactive entertainment products.

Video games consist of multiple integrated subsystems, each requiring specialized test approaches.

Testing in games often involves qualitative assessment. Components like difficulty balance, fun factor, narrative coherence, and emotional impact must be considered. Unlike typical software, minor functional defects may be tolerated, but poor gameplay leads to rejection by users.

Visual presentation is critical in games. Testers must evaluate animations, transitions, lighting, camera behavior, and VFX (visual effects) fidelity. Failures related to clipping, frame rate drops, or incorrect rendering can severely affect immersion.

Video games are typically released on multiple platforms (PC, consoles, mobile, VR (virtual reality)), each with its own hardware constraints and control schemes (e.g., keyboard/mouse, gamepads, motion sensors). Evaluating consistency and responsiveness across platforms is a core test objective.

Video games often support multiple languages and markets, requiring extensive localization testing. This includes not only translated text but also voice-overs, legal compliance, cultural sensitivity, and layout adjustments.

Video game testers must blend analytical test techniques with creative thinking. They often need to simulate player behavior, uncover unintended consequences of mechanics, and provide feedback on design coherence.

A common misconception is that game testers spend their time simply playing games. In reality, testers conduct structured testing based on defined test cases, specifications, and quality standards.

© International Software Testing Qualifications Board

While casual players engage for entertainment, testers systematically verify compliance with requirements, often replaying the same scenario multiple times to validate different aspects of the game.

Repetitive tasks and manual regression testing may seem monotonous, but identifying game-breaking failures is highly rewarding and crucial for delivering a polished product.

Connection with Other Syllabi

The approaches discussed in this syllabus apply exclusively to game testing. They are relevant to both software testing (e.g., testing video game mechanics) and hardware testing (e.g., testing controllers).

Some aspects of game testing are not significantly different from testing other software, especially when performing test types such as regression testing, performance testing, or usability testing. Core testing concepts, non-functional test types, and the specifics of testing software on mobile platforms are covered in detail in other ISTQB® syllabi.

1.2 Typical Roles in a Video Game Development Team

In most cases, a video game development team comprises various roles, many of which are similar to those found in other areas of software development. In smaller teams, one individual may take on multiple roles simultaneously. Below is a summary of the key responsibilities of typical roles in a game development project. Additional roles specific to certain areas of game development are covered in relevant sections of this syllabus.

Game Functionality	Role	Responsibilities
Story, video game mechanics, game design balance.	Game designer / Narrative designer	Directs the game's creative vision, designs storylines, video game mechanics, balance, reward systems, monetization strategies.
Graphics subsystem: 2D assets	Artist	Creates characters and assets, produces animation sprites.
Graphics subsystem: 3D assets	3D Modeler / Rigger / Skinner	Develops 3D character models and assets, creates skeletal rigs for animation, textures models.
Graphics subsystem: animation	Animator	Animates in-game objects and characters.
Graphics subsystem: level design	Level designer	Designs game level architecture, provides compliance with game rules at each level.
Graphics subsystem: user interface	UI (User Interface) designer	Develops graphic interfaces to enhance user interaction with the game and its subsystems.
Graphics subsystem: visual effects	VFX specialist	Creates various in-game visual effects.
Physics subsystem	Game engine programmer	Implements physics simulation within the game engine.

© International Software Testing Qualifications Board

Artificial Intelligence (AI) subsystem	AI programmer	Writes scripts to simulate realistic AI behavior for in-game characters.
Data storage subsystem	Server-side developer	Develops functionality for storing player data on servers.
Software subsystem: interfaces	Interface programmer	Implements user interfaces to facilitate intuitive interaction with the game and its systems.
Audio subsystem	Audio engineer	Records and processes game-related sound effects.
Audio subsystem	Composer	Composes and records background music and soundtracks for the game.
Game product as a whole	Producer	Manages game development from a business perspective.
Game product as a whole	Localization specialist, Consultant, Translator, Voice actor	Provides proper localization of the game product for specific markets.
Game product as a whole	Tester	Identifies defects across all game subsystems and evaluates the quality of the subsystems as well as the complete video game.
Game product as a whole	Business owner	Defines the business development strategy.

1.3 Test Activities Throughout the Game Software Development Lifecycle

The video game development lifecycle typically consists of several key phases:

Concept Phase

The main goal of this phase is to establish a unified vision of the future product between the development team and stakeholders. Testware created in this phase include:

- Test approach draft
- Test plan draft
- High-level test effort estimation
- Test environment outline

Pre-Production Phase

The primary objective of this phase is to develop a prototype—the initial working version of the game. The technical feasibility of the project is also assessed.

© International Software Testing Qualifications Board

Game testers and other development team members review the prototype to evaluate gameplay mechanics and identify early-phase technical challenges. Testware created and updated in this phase include:

- Test approach
- Test plan
- Test schedule
- Test environment setup

Production Phase

During this phase, the full game product is developed. This is the longest phase and is often divided into milestones for alpha and beta releases.

Testware created and updated in this phase include:

- Test plan
- Test schedule
- Test conditions, checklists, test cases, automated test scripts, test procedures, test suites
- Defect reports
- Test completion reports

Post-Production Phase

The post-production phase includes maintenance test activities that support the software development lifecycle. During this phase, updates and patches testing and regression testing is conducted, as described in [ISTQB_FL_SYL].

Live Operations

Live Operations (LiveOps) in video game development refers to the ongoing planning, delivery, and governance of post-release changes for a live game, with the objective of maintaining service continuity while evolving content, balance, and monetization. LiveOps typically includes time-bound events and seasonal content, server-side configuration and feature toggles, hotfixes, economy tuning, and controlled experimentation, all supported by operational monitoring and telemetry.

From a testing perspective, LiveOps requires a risk-based approach to change management: teams shall define release criteria, implement regression safeguards for critical player journeys, and verify backward/forward compatibility across client versions and platforms. LiveOps also encompasses incident response and service restoration.

© International Software Testing Qualifications Board

2 Testing Video Game Mechanics – 120 minutes

Keywords

Functional completeness, functional correctness, playtesting

Video Game Specific Keywords

Concept phase, client-side mechanics, core mechanics, gameplay mechanics, meta mechanics, non-gameplay mechanics, pre-production phase, production phase, server-side mechanics, video game mechanics

Learning Objectives for Chapter 2

2.1 Principles and Concepts of Video Game Mechanics

GaMe-2.1.1 (K2) Differentiate the testing of gameplay mechanics and non-gameplay mechanics

GaMe-2.1.2 (K2) Differentiate the testing of core mechanics and meta mechanics

GaMe-2.1.3 (K2) Differentiate the testing of client-side, server-side, and client-server mechanics

GaMe-2.1.4 (K2) Give examples of failures in video game mechanics

2.2 Video Game Mechanics Testing

GaMe-2.2.1 (K3) Apply test types and test approaches to video game mechanics testing

2.1 Principles and Concepts of Video Game Mechanics

Mechanics are the foundation of any game. They serve as a means of interaction between the game and the user. Mechanics take into account the influence and feedback between the game and the user, as well as changes in the video game state within predefined requirements.

Video game mechanics largely determine the player's experience, impression, and level of engagement. Since video game mechanics are at the core of any game, verifying their correct operation is typically a top priority for testers, though priority should be determined through risk-based testing considering business impact, player experience, and technical dependencies. Failures found during such testing are typically among the most critical.

A defect in a core video game mechanics, causing the failure of a jump action, can make it impossible to overcome obstacles and, consequently, complete the game.

Even if video game mechanics are rarely used or are unrelated to the game's progression (e.g., a character's ability to ride a horse), their incorrect operation can negatively impact the overall perception of the game.

Video game mechanics must not only function correctly but also remain consistent and intuitive throughout the game. Inconsistent behavior, such as variable jump height or unpredictable collision detection, can frustrate players and break immersion. That's why testers often develop specific scenarios to validate edge cases and evaluate if the mechanics behave reliably under different conditions.

© International Software Testing Qualifications Board

Moreover, mechanics often interact with each other — for instance, combining a sprint function with a climbing mechanic. Testing these interactions is crucial, as defects may not surface during isolated tests but appear when features are combined. Automated test tools are used to verify functional correctness and performance efficiency, while playtesting assesses how mechanics feel and function in real gameplay from a player experience perspective. Ultimately, providing smooth and reliable mechanics significantly enhances user satisfaction and gameplay fluidity.

Types of Mechanics in Video Games

Mechanics can be classified based on different aspects and objectives of the game:

- Based on the type of player interaction with the mechanics:
  - Gameplay Mechanics
  - Non-Gameplay Mechanics
- Based on their influence on core gameplay:
  - Core Mechanics
  - Meta Mechanics
- Based on game architecture:
  - Client-Side Mechanics
  - Server-Side Mechanics
  - Client-Server Mechanics

2.1.1 Differences Between Testing Gameplay and Non-Gameplay Mechanics

Testing gameplay and non-gameplay mechanics is a part of functional testing and is conducted using test approaches, test techniques, and tools described in [ISTQB_FL_SYL]. However, there are key differences in how they are tested.

Gameplay Mechanics

Gameplay mechanics are elements that directly influence player progress toward game objectives and form the core interactive experience. These mechanics may be directly controlled by the player (e.g., character movement, shooting) or may affect gameplay indirectly (e.g., enemy AI, environmental hazards). While many gameplay mechanics provide visible feedback, some operate with hidden calculations (e.g., damage formulas, random generation).

Testing gameplay mechanics usually has a higher priority than testing non-gameplay mechanics because they are more noticeable to players. Failures in these mechanics are likely to be observed and can lead to negative feedback from the gaming community.

Gameplay mechanics require verification of functional correctness and functional completeness and validation of functional appropriateness. In addition to functional testing, usability and UI testing are also conducted.

A tester can often predict how gameplay mechanics should function and its expected results, even if documentation describing the mechanics is incomplete.

Non-Gameplay Mechanics

Non-gameplay mechanics are typically hidden from the user, although they may be triggered by player actions. For example, an in-game purchase may initiate the following sequence of non-gameplay mechanics:

1. A request is sent to the database to retrieve account data.
2. A transaction is performed to deduct funds.

© International Software Testing Qualifications Board

3. The purchase is logged.
4. A confirmation email is sent to the user.
5. The purchase date and the item type are recorded for future promotional offers.

Testing non-gameplay mechanics primarily focuses on verification of functional correctness and functional completeness. While failures in these mechanics may not directly affect the player, they can distort game analytics, leading to incorrect business decisions and potential player churn.

Unlike gameplay mechanics, testing non-gameplay mechanics requires detailed requirements specifications or predefined test cases, since they are often implemented at the developer's, publisher's or marketplace request and vary across games.

However, gameplay mechanics include factors like latency, fairness, and exploit resistance, in addition to responsiveness, among other non-functional parameters. So functional testing is just one aspect of mechanics testing.

2.1.2 Differences Between Testing Core and Meta Mechanics

Core Mechanics

Core mechanics define the fundamental game functionalities and are responsible for providing players with the intended experience. They categorize a game into a specific genre.

Examples of core mechanics:

- Jumping (Platformer game)
- Building structures (Real Time Strategy game)
- Shooting (First-Person Shooter game)

Removing a video game core mechanics can entirely change the game's genre. For example, removing vehicle movement from a racing simulator would eliminate its classification as a racing game.

Meta Mechanics

Meta mechanics are gameplay elements that enhance the gaming experience but are not essential to defining the game's genre. Removing a meta mechanics does not change what type of game it is, though it may reduce player engagement or variety.

Examples of meta mechanics:

- Choosing a car model based on track type in a racing game - not a mandatory action, but it can increase the chance of winning a race
- Achievement tracking systems - provide additional goals but don't change core gameplay
- Collecting collectible items - increases player retention but does not affect the core gameplay

A regular energy replenishment in a match-3 game - motivates the player to log in more often.

Testing Core and Meta Mechanics

Core mechanics often have stricter performance efficiency requirements because delays can directly impact gameplay objectives and player success. For instance, if weapon switching in combat (core mechanics) is slow, a player may lose a battle.

In contrast, the time behavior of meta mechanics is usually less critical. For example, a slight delay in changing a character's appearance may be classified as a low-priority defect, since it doesn't affect the ability to achieve game objectives.

© International Software Testing Qualifications Board

The distinction between core and meta mechanics also influences how defects are prioritized during testing. Since core mechanics directly affect gameplay flow and the ability to achieve game objectives, defects in these systems often require immediate attention and validation. On the other hand, defects in meta mechanics, while still important for overall user satisfaction, may be deprioritized if they do not disrupt critical gameplay functions. This prioritization provides that development and test efforts align with the player's most essential interactions, maintaining a smooth and engaging experience.

The differences between core and meta mechanics also impact the timing of their testing. For example, core mechanics are typically tested earlier and more frequently throughout development due to their critical nature. More details on testing at different development phases are discussed in the next section .

2.1.3 Differences Between Testing Client-Side, Server-Side, and Client-Server Mechanics

The need to test client-side, server-side, and client-server mechanics depends on the game's architecture. Each type of mechanics requires different test approaches based on its implementation.

Client-Side Mechanics

Client-side mechanics are processed on the player's device, meaning all relevant data is stored locally. Some games function entirely on the client side, eliminating the need for a server. Such games are typically single-player and do not require an internet connection.

Even in multiplayer games, certain mechanics remain client-side, such as viewing the game map or opening the inventory. These are tested by applying functional testing without server interaction.

Client-side mechanics are generally tested using black-box testing, often involving UI testing.

Server-Side Mechanics

Server-side mechanics run exclusively on the game server, which protects critical game logic from player manipulation.

Testing server-side mechanics is especially important for multiplayer online games. For example, in player-versus-player PvP matchmaking, the server provides fair opponent selection based on player rankings. If this process fails, it could lead to an imbalanced and frustrating gameplay experience.

Unlike client-side testing, server-side mechanics are tested using server consoles and database queries rather than UI interactions. Testers often analyze logs and database transactions.

Key test types for server-side mechanics include:

- Functional testing
- Performance testing
- Security testing

Client-Server Mechanics

Client-server mechanics combine aspects of both previous types. One part of the logic is processed on the player's device, while another is handled on a remote server. These mechanics continuously interact, exchanging data. In multiplayer games, the server receives data from the client and then distributes the results of actions to all affected players.

Data received by the server undergoes mandatory validation. For example, if a player attempts to make a purchase in the in-game store, the server must verify the availability of sufficient in-game currency

© International Software Testing Qualifications Board

before completing the transaction. Even if a player modifies the client-side data to gain an unfair advantage, the server will replace it with the correct values.

Scenario-based testing can be used to test client-server mechanics. Testers create one or more test cases to cover as many different mechanics as possible.

2.1.4 Types of Video Game Mechanics Failures and Their Possible Causes

Regardless of their type, game mechanics are prone to the following types of failures:

- Mechanics functional failures
- Mechanics feedback failures
- Mechanics efficiency failures in specific game environments

Mechanics Functional Failures

This type of failure is directly related to the functional suitability of video game mechanics. As with non-gaming software, functional failures occur when a specific feature behaves incorrectly.

For example:

- Weapons do not reload
- Resources cannot be collected
- The zoom function does not work when using binoculars

These failures typically arise due to defects in the game code.

Mechanics Feedback Failures

Mechanics feedback failures are associated with a lack of proper response from the game when video game mechanics are used. For instance, a problematic situation occurs when a player does not understand why the game has ended. This is why video game mechanics are often accompanied by visual and sound effects. These could include explosion animations, a countdown timer sound, or even a text notification.

Testing evaluates whether a response is present when using the mechanics and evaluates whether its presence or absence is justified. The correctness of the effect itself is verified through other types of testing, such as graphics and audio testing.

Mechanics Security Failures

These are failures in game rules and mechanics that enable intentional misuse to compromise game-state integrity or the in-game economy, obtain an unfair advantage, or harm other players and the live service. Typical failures include: missing invariants and insufficient server-side validation for critical events (e.g., rewards, progression, trading); authority flaws where the client can improperly influence outcomes; incorrect handling of edge and failure states (e.g., cancel/retry flows, disconnects, duplicate event delivery); imbalance-prone farming loops; item or currency duplication; bypassing constraints (e.g., cooldowns, limits, access requirements); manipulation of matchmaking or ranking; and inadequate observability (insufficient logs/telemetry to support detection and investigation).

Testers shall classify such failures using risk-based criteria (impact on fairness, revenue/monetization, reputation, and account security) and define acceptance criteria covering: server-side enforcement, atomicity and idempotency of state-changing operations, resilience to replay and fault scenarios, effective limits and abuse controls, and demonstrable absence of paths leading to illegitimate player power gains or economy destabilization.

Mechanics Efficiency Failures

© International Software Testing Qualifications Board

The third type of defect occurs when a mechanics functions correctly in isolation and provides the expected information to the player, but is not compatible with the other video game mechanics with which it interacts. Such a defect may manifest as an inconsistency with comparable mechanics, an imbalance in effectiveness (overpowered or underpowered) during gameplay, or any situation in which the mechanics behaves as intended individually but introduces a system-level defect when integrated into the overall game system.

For example, a game designer might add two opponents with different behaviors and characteristics. One is fast and agile but deals low damage, while the other is slow and clumsy but delivers powerful attacks. Both opponents have their strengths and weaknesses and are designed to present similar challenges to the player. However, if most of the game level consists of high platforms that the player's character can climb, the balance changes. The player may safely attack the slow enemy from above, making it less of a threat. Meanwhile, the fast, high-jumping opponent can still pose a significant challenge.

Detecting such failures is difficult because predicting these situations and how the mechanics will behave within them is inherently challenging.

2.2 Video Game Mechanics Testing

2.2.1 Applying Fundamental Approaches to Video Game Mechanics Testing

Video game mechanics testing can be performed at various phases of software development, using different test approaches and test types.

Concept Phase

At the concept phase, documentation describing the gameplay mechanics and the rules of their operation are reviewed. It is important to understand that some mechanics may be conceived, described, and consequently reviewed at later phases.

Pre-production Phase

During the pre-production phase, only the core mechanics of the game are implemented. The primary tasks of a tester at this phase include verifying mechanics based on the following characteristics:

- Functional correctness and functional completeness of the mechanics.
- Functional appropriateness of the mechanics to players.

Used Test Approaches:

- Rapid prototypes: Creating simplified versions of mechanics for quick evaluation of ideas.
- Game mockups: Testing key mechanics without complex visual components.
- Iterative approach: Gradually improving mechanics based on collected data and feedback.

Production Phase

During the production phase, implemented mechanics undergo testing to verify they meet the specified requirements.

For testing the interaction between different mechanics, exploratory testing and error guessing are particularly effective. In large-scale multiplayer games, the number of possible interactions becomes extremely large and difficult to predict. Exploratory testing allows testers to use their creativity and domain knowledge to discover unexpected interaction defects, while error guessing leverages experience to identify likely problem areas. These experience-based test techniques complement scripted testing by uncovering defects that predefined test cases might miss.

© International Software Testing Qualifications Board

Manual playtesting with various participant groups is conducted for different purposes:

- Internal playtesting: Testers and developers play to identify functional defects, balance defects, and usability defects.
- External playtesting (Early Access, Open Beta Testing): Real players play while being observed to assess user experience, identify confusion points, and gather feedback.
- Focus groups: Facilitated discussions with players to gather qualitative feedback on mechanics, appeal, and perceived balance.
- A/B testing: Players are randomly assigned to different mechanics versions, with quantitative metrics compared to determine effectiveness.

When testing with a large number of players, meta mechanics are evaluated, along with the usability and clarity of the game for users.

The main objective at this phase is to gather player feedback and assess the impact of implemented mechanics on key metrics such as:

- Average session length
- Number of invited friends
- Average spending per player
- Most popular in-game items.

Used Test Approaches:

- Mathematical models: Analyzing game design balance by using formulas and algorithms.
- Statistical analysis: Collecting gameplay data (e.g., win/loss ratios, difficulty levels).

Post-Production Phase

After the game is released to the market, testers are responsible for identifying and verifying defects in mechanics reported by players, as well as testing new mechanics introduced into the game. Canary testing and A/B testing can be used in this case. Video game mechanics testing uses a combination manual testing and automated testing. Automated testing is effective for verifying functional correctness (e.g. damage calculations, state transitions), while manual playtesting is essential for assessing subjective qualities like game feel, balance, and player experience. The appropriate mix depends on the mechanics type and test objectives.

© International Software Testing Qualifications Board

3 Video Game Level Testing - 120 minutes

Keywords

Geometry testing,playtesting

Video Game Specific Keywords

Block-out, color coding, game character, grey box, setting, structural geometry, terrain, texture, video game level

Learning Objectives for Chapter 3

3.1 Principles and Concepts of Level Design in Video Games

GaMe-3.1.1 (K2) Distinguish the video game levels components

GaMe-3.1.2 (K2) Give examples of failures in video game levels.

3.2 Test Types in Video Game Levels Testing

GaMe-3.2.1 (K2) Summarize geometry testing.

GaMe-3.2.2 (K2) Summarize the playtesting.

3.3 Executing Video Game Level Testing

GaMe-3.3.1 (K3) Apply level testing.

3.1 Principles and Concepts of Level Design in Video Games

3.1.1 The Term "Level" and Its Specifics

A video game level represents a defined environment or scenario in which the player interacts with video game mechanics to achieve one or more gameplay objectives. Depending on the game genre and design, a level may range from a simple, self-contained space to a large, complex environment supporting multiple activities and player behaviors.

Levels in games consist of multiple components combined into a cohesive whole. The primary components include:

- Structural geometry
- Game Environment Objects
- Textures
- Lighting, Reflections, and Shadows
- Sound Effects
- Gameplay Functionality

Structural Geometry

Structural geometry, or the foundational geometry of game levels, is one of the most crucial components that define any in-game surface. It establishes the terrain on which game characters can move. Structural geometry also limits the playable area of a level. If a game level has clearly designated entry and exit points, structural geometry can guide the player toward the level's completion. Depending on the game's setting, the components forming a game level may vary.

© International Software Testing Qualifications Board

Game Environment Objects

Game environment objects are models or groups of models used to create a more realistic depiction of the necessary game space.

Static objects complement and define the terrain or structural geometry of a level. Dynamic objects are interactable elements that are used either directly within gameplay mechanics or to support immersion and the overall player experience.

Textures

All game level objects are covered with textures that simulate different surfaces and materials. These textures can represent components such as walls, floors, terrain, and materials like wood, metal, and stone. They may also include details like footprints, scratches, or dirt. The level of detail and realism in textures significantly influences the player's immersion in the game world.

Lighting, Reflections, and Shadows

The use of different light sources helps create the desired atmosphere in a game level, guide the player’s movement, draw attention to specific areas, or hide them from enemies.

Sound Effects

Background music and sound effects accompany various actions and events within the game.

Gameplay Functionality

Gameplay functionality consists of various settings that organize the gaming process on a particular level. These settings include:

- Settings related to level progression:
  - Player and enemy spawn points
  - Win and loss conditions
  - Automatic save points
  - Scripted events triggering specific game actions or sequences (e.g., an airplane flying overhead every two minutes, a cutscene playing when a character enters a building, or an alarm activating upon entering a restricted area)
- Settings for interactive objects such as doors, vehicles, destructible objects, and traps
- Physical boundaries preventing the character from leaving the level or getting stuck in its geometry

Each component of a game level may contain defects. Defects affecting lighting, shadows, and reflections are typically identified through graphic system testing, while defects in sound content are detected during audio testing.

3.1.2 Types of Video Game Level Failures

There are several types of failures that commonly occur during the creation of video game levels. These failures can be difficult to detect and may remain unnoticed for some time. In most cases, the best way to identify level failures is through testing by experienced players who can recognize failures and report defects.

A significant portion of failures found in video game levels can be grouped into several common categories, each affecting gameplay, navigation, or player perception in different ways.

© International Software Testing Qualifications Board

1. Navigational and structural failures affect player movement and access within the level and include failures such as incorrect object scaling, stuck points, map holes, inaccessible areas, and invisible walls.

Incorrect Object Scaling

This type of failure includes objects on the map that are improperly scaled and may be used by players as cover. For example, a player may be unable to take cover behind a wall or a crate even when crouching. Such failures can alter the entire gameplay experience and/or provide an unfair advantage to certain players.

Stuck Points

A game character may become stuck in the level's geometry, unable to escape, or may enter restricted areas. Identifying such failures is especially important in multiplayer games, where level design defects can provide an unfair advantage to certain players or teams.

Stuck-related failures can be caused by various factors. One common cause is level reconstruction, which may disrupt the navigation mesh (a data structure defining traversable areas for NPC pathfinding, covered in detail in Chapter 6) or its automatic generation. For example, if models are repositioned during level reconstruction but the navigation mesh remains unchanged, non-player characters (NPCs) may attempt to move through walls or other objects they cannot bypass, resulting in them getting stuck—a situation commonly referred to as a stuck point.

Map Holes

Another common failure is map holes. These failures result from incorrect level geometry, where parts of the level model (such as walls and floors) do not fit tightly together. As a result, a character may fall through the map, entering an area with no support, causing them to endlessly fall into the void due to gravity.

Inaccessible Areas

In video games, especially in multiplayer maps, there may be intentional locations where a player gains a tactical advantage over others. For example, a machine gunner positioned on a hill may have a wide firing sector. While the first player to reach such a position gains an advantage over opponents, this is a deliberate design choice and not a failure.

However, there are cases where such areas are unintentionally created due to level design defects. The developer may have intended a specific area to be inaccessible, but a certain character, utilizing unique abilities (such as higher jumps), may manage to reach it. As a result, the player may gain an unfair gameplay advantage over opponents, such as being undetectable or immune to attacks.

Similar failures may also include the absence of a notification for a player when going out of bounds.

Invisible Walls

These failures are usually the result of an incorrect object modeling. They often occur when the visible model of an object does not match its collision model—the physical model that defines the object's tangible boundaries.

2. Visual and presentation failures impact immersion and clarity and include floating objects, incorrect color coding, texture seaming, and failures related to the consistency of objects with the game's established theme and context..

Incorrect Color Coding

© International Software Testing Qualifications Board

Objects encountered by the player during gameplay have different properties. Some of them serve as decorations, while others are interactive. Color coding of objects with different properties is an essential aspect of level design in video games. Therefore, once a specific color coding is established, it should remain consistent throughout the game. Similarly, breakable crates, doors that can be opened, climbable rocks, and ledges should be visually distinct in shape and/or color from other objects.

If color coding is inconsistent, players may waste time figuring out which objects are interactive and what actions they need to take to progress.

Texture Seaming

Such failures are typically noticeable on large maps and locations where different surface textures, such as sand and grass or sand and water, blend together. The transition between textures may not always appear seamless and natural.

Real (Historical) Prototypes Recognizability

In games set in worlds that mimic reality, it is crucial to maintain resemblance to recognizable objects. These may include architectural landmarks, landscapes, and geographic locations. Historical accuracy must also be preserved, as its absence can disrupt player immersion.

Floating Objects

During level development, multiple specialists may work on the same location simultaneously. Some of them may add or remove objects, while others modify the terrain geometry by raising or lowering sections of the map. As a result, failures may occur where previously grounded objects become suspended in mid-air.

Unnecessary Gameplay Complications

These failures arise when there is a lack of clear objective markers or when required actions to progress through a level are not intuitive. For example, a player may defeat all enemies in the accessible area but remain uncertain about the next step needed to advance.

Artificial Restrictions and Constraints

These failures do not impact gameplay balance or provide unfair advantages but negatively affect immersion and player perception. For example, a player may encounter a waist-high fence blocking their path but find it impossible to climb over. While the level designer may have intended this area to be inaccessible, from the player's perspective, such a limitation feels like an oversight that disrupts immersion. Similarly, a player may perceive it as a defect when a character is trapped behind a metal grate, unable to escape, despite the gaps between the bars being wide enough to pass through.

Level Balance

Many failures arise from improper level balance adjustments. For example, overly difficult enemy characters at a given phase of the game may make level completion impossible. Such failures are often identified during AI testing. In multiplayer games, where multiple teams compete to reach a designated point on the map, level balance failures may include unfair starting conditions that give one team an advantage over another.

Narrative Inconsistencies

Level design failures disrupt the overall storytelling style or game narrative. While these failures have minimal impact on core gameplay mechanics, they negatively affect player immersion and the overall perception of the game. For example, if a character arrives in a city that, according to the story, was

© International Software Testing Qualifications Board

devastated by a nuclear bombing, the presence of undamaged buildings would create a narrative dissonance.

Destructible Environment Failures

If the game environment is non-interactive — meaning the player cannot interact with it — the level of realism remains low, even if environmental objects are accurately modeled. The concept of a "destructible environment" encompasses various components such as terrain (referred to as "deformable terrain"), vegetation, interiors, buildings, and other structures. Special attention is given to destructible environments in games where they play a crucial role in gameplay or even become part of core video game mechanics.

A destructible environment can also be integrated into the storyline. For example, the destruction of a wall might block the player's path, forcing them to find an alternative route pre-programmed by the developers.

3. Performance-related failures occur when level design or content causes unacceptable degradation in frame rate, loading times, or system responsiveness. Common causes include overly dense geometry, duplicated or overlapping meshes, excessive use of high-resolution textures, inefficient lighting setups, or poorly optimized environmental effects

3.2 Test Types in Video Game Levels Testing

Considering the characteristics and development sequence of game levels from geometric prototyping to the final refinement of details and objects it is logical to distinguish two types of video game level testing:

- Geometry Testing (including texturing)
- Playtesting

Testing at these levels does not consist of isolated activities. The division is based on the need to focus on identifying defects with a common root cause. This approach not only simplifies defect detection but also confirms that defects are properly addressed for subsequent fixes.

3.2.1 Geometry Testing

Testing of game levels begins at the earliest phases of their creation. The test objects, the nature of tests and the order of test execution, as well as the responsibilities of various specialists, depend on the current phase of level development.

Block-Out/Grey Box

Based on a previously developed sketch, in 3D video games a three-dimensional model of the future game level is created using special objects. Only components that directly impact gameplay are used: grey cubes, spheres, cylinders, and planes that schematically form the layout of the game level.

In this phase, not all video game mechanics may be fully implemented, requiring the level designer to gradually integrate them while adapting the level to the constantly evolving development conditions.

To create a prototype with proper proportions and scaling, the level designer collaborates with the game designer to define a set of measurements and parameters for the game character. These measurements directly influence the size of objects and environmental components within the level.

Once the level designer has determined the appropriate scale and size for the game level and its objects, additional functionality is configured to support gameplay. This includes spawn points for characters, scripted events, and other interactive components.

© International Software Testing Qualifications Board

Final Version

The level designer integrates the software components created by the entire development team into a cohesive whole. At this point, decorative details are added to the game level, and objects begin to resemble real-world counterparts not only in appearance but also in function and placement. During the development of the level geometry, it is essential to conduct functional and visual validation tests focusing on:

- Object proportions and dimensions within the game level, providing consistency with game design documents and visual expectations.
- Level boundaries, to verify they behave as expected (e.g., collision boundaries, invisible walls, playable area limits).
- Texturing fidelity, which includes correct mapping, alignment, and resolution consistency across surfaces.

These tests help detect visual inconsistencies, clipping defects, and environmental design defects, such as map holes, contributing to both playability and immersion quality.

3.2.2 Playtesting

The test objective of this type of video game level testing is to identify defects that are not directly related to its geometry. This process involves evaluating all aspects that enhance player immersion and improve overall gameplay convenience.

Key focus areas of such testing include:

- Overall level balance
- Absence of inaccessible areas or logical justification for their inaccessibility
- Clear and intuitive navigation and level orientation
- Elimination of unnecessary complexity and artificial constraints
- Consistency of objects with the game's established theme and context.
- Absence of invisible and floating objects
- Environmental destructibility (if applicable)

Since game levels are typically vast spaces, designing detailed test cases to verify all of the above can be challenging or even impractical. Crowd testing can be an appropriate test approach in this case.

In addition, testers should pay attention to how the level supports different gameplay styles — for example, stealth versus aggressive approaches. The placement of cover, interactive components, and resources should support these styles without giving unfair advantage to one. Sound design and ambient cues also play a significant role in level perception and should be validated as part of the overall experience. Player feedback gathered during exploratory testing is especially valuable for identifying subtle defects that formal test plans might miss.

3.3 Executing Video Game Level Testing

3.3.1 Applying Fundamental Approaches to Video Game Level Testing

Earlier, different phases of level creation were discussed, from geometric prototyping to the final version. From a testing perspective, the varying degrees of completeness of a level determine which test approaches are appropriate and what aspects of the level can be meaningfully evaluated at each phase. Test activities should therefore be selected and applied based on what is implemented and stable, rather than treating all phases as equally testable.

© International Software Testing Qualifications Board

While playtesting is performed throughout level development, its focus evolves as the level matures. Early playtesting emphasizes basic layout, navigation, and mechanical feasibility, whereas later playtesting evaluates gameplay flow, balance, performance, and overall player experience.

Testing the Prototype of a Video Game Level

Since requirements are not yet finalized at the prototyping phase, exploratory testing and other experience-based test techniques are considered the most effective.

Initial playtesting is conducted on the created prototype to assess the gameplay on this map. Participants in this early playtesting may include testers from the game development team as well as external testers. By gathering feedback from playtest participants, developers can evaluate successful and unsuccessful design choices and make necessary adjustments, relocations, or removals of objects within the prototype.

Additionally, playtesting of the prototype evaluates video game mechanics and their interaction with object placement and size within the level layout.

For example, cover components on the game level must be large enough for the character to hide behind them, surface gaps must be appropriately sized for the character to jump across, and ledges must be accessible for climbing without the character hitting their head on low ceilings.

Furthermore, testing evaluates if the level layout creates the intended game difficulty and provides the right gameplay experience. For instance, if a section of the game is designed to feature a firefight with multiple enemies, the level must contain a sufficient number of cover points. Conversely, if the level includes a battle against a single powerful and resilient enemy (a boss fight), the playable area should be constrained and cleared of unnecessary objects to enhance the intended challenge.

Prototype Geometry Testing

In this phase, testing validates the actual results obtained by the artistic rendering of the game level that could interfere with gameplay.

Examples of such failures include a temporary 3D model of a rock blocking the level’s passage or a tree canopy obstructing the player's view.

Additionally, tests are performed to evaluate that there are no gaps between essential level components, such as the "floor" and other objects, to prevent unintended voids in the map.

To identify these failures, playtesting is conducted again, focusing on evaluating video game mechanics concerning the size and placement of objects according to required metrics and providing the correctness of gameplay.

Simultaneously, additional testing may be carried out for object appearance, lighting, game environment, and other visual aspects.

Final Version Testing

As part of playtesting, testers verify the presence of collision models for all game objects, determining physical and visual object models correct alignment.

In this phase, performance testing and compatibility testing are conducted. The functional suitability of the game level is tested on various devices, frame rate measurements are taken, and the quality of object rendering at different distances, lighting, and shadows is verified.

Collision model testing can be performed as follows: the tester moves the game character toward an object, such as a large rock, and attempts to "pass through" it from different angles. If the rock's collision

© International Software Testing Qualifications Board

model is properly configured, the character's body will only visually touch the rock without passing through or becoming embedded in it.

In this phase, essential tests include verifying the correctness of object texturing (no visible "seams") and confirming the alignment of objects with real (historical) prototypes.

© International Software Testing Qualifications Board

4 Graphics Testing - 95 minutes

Keywords

Artistic testing, graphics technical testing, playtesting

Video Game Specific Keywords

3D model, animation, clipping**Ошибка! Закладка не определена.**, collision, hitbox, level of detail (LoD), mapping, occlusion, scene lighting, texture, visual effect

Learning Objectives for Chapter 4

4.1 Principles and Concepts of Video Game Graphics

GaMe-4.1.1 (K1) Recall the types of video game graphic components

GaMe-4.1.2 (K2) Give examples of failures in video game graphics

4.2 Test Types in Video Game Graphics Testing

GaMe-4.2.1 (K1) Recall video game graphics artistic testing

GaMe-4.2.2 (K1) Recall video game graphics technical testing.

GaMe-4.2.3 (K1) Recall video game playtesting.

4.3 Executing Video Game Graphics Testing

GaMe-4.3.1 (K3) Apply graphics testing

© International Software Testing Qualifications Board

4.1 Principles and Concepts of Video Game Graphics

4.1.1 Features of Video Game Graphics

Any gaming product consists of numerous graphic objects, collectively enabling users to visualize the gameplay experience.

Models

A model is an object within computer graphics. Based on image creation methods, graphic models can be categorized as follows:

- Two-dimensional computer graphics (2D)
- Three-dimensional computer graphics (3D)
- Computer-generated imagery (CGI)

Model Animation

It is the process of making 3D characters, creatures, or objects move and behave in a lifelike or stylized way. It involves applying motion to digital models so they can walk, run, talk, gesture, or interact with their environment during gameplay.

Model Collision

Models within a video game scene interact with one another. Any interaction begins with contact (collision). To enable a model to interact with its environment and other objects, a collider is created—an invisible simplified shape attached to the object to calculate collisions with other objects. The collider can be considered the physical model of the object. While it remains invisible to the player, any collisions with colliders are processed by the game engine.

However, some objects in the video game may serve purely decorative purposes and may not have any collision detection at all.

Model Realism and Authenticity

In addition to entertainment, video games can often include an educational component. In such cases, maintaining realism and adherence to real (historical) prototypes in the use of graphic assets becomes particularly important.

Inaccurate details can not only distort the perception of history but also undermine the developers' efforts in crafting an authentic atmosphere.

Model Textures

In a broader sense, textures can be considered as patterns on the surface of a sculpted object. Using textures allows for the reproduction of fine surface details that would be excessively complex to create with polygons, such as scars on the skin, wrinkles on clothing, small rocks, and other fine details on walls and terrain surfaces.

Visual Effects (VFX)

Visual effects can be categorized into two main types:

- Gameplay Effects
- Environmental Effects

VFX consist of a combination of animations, lighting effects, particles, and other graphic objects that make the video game world more realistic, dynamic, and engaging.

© International Software Testing Qualifications Board

Scene Lighting, Reflections, and Shadows

Scene lighting is used to create different moods for the player, alter their perception of in-game events, and encourage specific behaviors. Thus, many lighting techniques used in visual arts, cinema, and architecture are also applied in video games to enhance the aesthetics of virtual spaces and improve player immersion.

4.1.2 Types of Failures in Video Game Graphics

The classification of failures related to the graphic subsystem of a video game is based on and directly linked to the areas of this subsystem mentioned in the previous section. [Torgovkin24]

Types of Software Failures in Video Game Graphics

- Model failures
- Texture mapping failures
- Collision failures
- Occlusion Culling Failures
- Hitbox failures
- Realism and authenticity failures
- Animation failures
- Animation transition failures
- Dismemberment gore system failures
- Camera shake failures

Model Failures

To better understand the root causes of defects related to failures that arise in the graphic subsystem of a game, as well as the verification process, it is essential to first examine how graphic objects are created and how they interact with each other.

When working with 2D models, the verification process in the early phases is relatively straightforward: the object should be evaluated for historical accuracy (if it has a real-world prototype) and how different users perceive it during localization. However, 3D models introduce additional complexities.

A 3D model is composed of polygons—multi-sided shapes whose corner points have precise coordinates in three-dimensional space and are connected by lines. The higher the level of detail, the greater the number of polygons required for modeling.

It is important to recognize that calculating the movement of a large number of points during model animation (especially when many such models are present in a game scene) places a significant load on the hardware of the gaming platform. This, in turn, substantially increases the performance requirements of the graphics processor (GPU).

As a result, high polygon counts reduce the potential player base for the game by limiting accessibility to users with lower-end hardware. This can significantly impact the game's commercial success.

Thus, one of the key tests in graphics technical testing is verifying that the model is optimized for use within the game engine and will not cause performance failures. These tests are conducted regularly throughout development to enable continued optimization.

Texture Mapping Failures

In the phase of applying textures to the created model, various failures may arise, including:

- Incorrect texture scaling, causing distortion of the image on the surface of the modeled object.

© International Software Testing Qualifications Board

- Visible seams between textures covering different parts of the model.
- Selection of low-quality, low-resolution textures, reducing the visual appeal.
- Incorrect transparency settings, as well as misconfigured reflection and refraction parameters.
- Repetitive texture tiling on a single surface, significantly reducing the model's realism.

The primary aspect of texture testing is verifying their presence on visible objects, providing correct display and uniformity. If the video game uses high-resolution textures, there should be no low-quality textures disrupting the overall visual consistency.

Collision Failures

All model interactions occur through collisions. Most models in a video game include a collider, and the size and configuration of this collider play a critical role in the emergence of collision failures.

If an object's collider is much smaller than its visible model, the character may "clip through the texture" or walk through it. Strictly speaking, the phrase "clip through the texture" (for example, when a character can pass through another object) is technically incorrect but accurately describes what the player sees when their character is partially or fully submerged inside another object.

These failures are especially critical in multiplayer games, as they can provide an unfair advantage to certain players. This is why testers must pay special attention to objects on PvP maps to evaluate if there is a fair gameplay.

If a collider is larger than the visible model, situations can arise where the character runs into an invisible wall or stands on an invisible platform [Buttfield19]. These failures are often exploited for speedrunning, where players deliberately search for areas where developers forgot to properly adjust colliders. In some cases, this allows players to bypass large sections of a level and significantly reduce completion time.

Occlusion Culling Failures

Failures caused by incorrect occlusion culling algorithms result in the display of scene components that should be hidden.

If one opaque object partially covers another; the obscured part of the second object should not be visible. Ideally, such areas are excluded from the rendering process to optimize computational resources. However, if the rendering engine's occlusion culling algorithm is incorrectly implemented or configured, the user may see object components that should remain hidden.

Hitbox Failures

Character interaction in games often occurs aggressively, meaning they involve damage infliction. Therefore, a hitbox is included in the collider model.

A hitbox is a part of the physical model, which registers a collision with a programmed object (e.g., a bullet, spear, sword, or fist) as damage dealt. In other words, if an object designed to deal damage collides with the hitbox of another object's physical model (collider), a certain number of health points is subtracted from the second object's health variable. To maintain realism, a model—such as that of a human-like creature—may have multiple distinct hitboxes to simulate varying levels of damage. Therefore, the tester’s task is to verify the correct placement of hitboxes and the accuracy of damage application when interacting with them, as defects in this area can lead to significant player frustration and dissatisfaction.

Realism and Authenticity Failures

Failures related to real-prototype accuracy occur when the visual or behavioral attributes of gameplay objects, based on real characters, items, or events, are significantly distorted. These inconsistencies

© International Software Testing Qualifications Board

may include incorrect proportions, colors, animations, or details that compromise historical or cultural authenticity. Such defects negatively impact the realism and immersion of the game world, particularly in simulation or historically themed games, and may lead to dissatisfaction among players and public criticism.

Animation Failures

Failures related to model animation can arise due to incorrect operation of the skeletal joint system, which may manifest as chaotic or unnatural limb movements.

Similarly, failures related to facial animation may occur, appearing as a complete lack of animation, an unnatural transition from one facial expression to another or a facial animation and the contextual situation mismatch.

A specific case of animation failures is the lack of lip sync, where lip shapes corresponding to specific phonemes are missing, incorrect sequences of lip shapes are used or animation is entirely absent.

Animation Transition Failures

Since a model can use different animations for performing various actions (idle, walking, running, jumping), various failures may occur.

A common failure related to animation transitions is the so-called "sliding" effect. This occurs when a character abruptly changes movement direction, such as suddenly turning and running to the side after moving forward. Such failures are especially noticeable in third-person games and often arise due to an insufficient number of animations for smoothly displaying different speeds and movement directions.

Dismemberment Gore System Failures

The effect of dismemberment is implemented by developers to simulate the detachment of individual limbs or the complete dismemberment of a character's body.

To achieve this effect, the model must be pre-prepared by "dividing" it into smaller parts. The effect is triggered when multiple conditions are met simultaneously:

1. The character must have less than a specified amount of health points (HP).
2. The damage must be inflicted by a specific type of weapon.
3. A shot from this weapon must hit a specific area of the character’s model.

When the specified conditions are met, the effect is triggered, resulting in the detachment of a part of the model or its "explosion," scattering fragments and adding visual effects such as blood and other details.

Failures that may occur include unnatural detachment of limbs, remaining parts of the model staying in place after the "explosion" and/or absence of the intended visual effects.

Camera Shake Failures

The effect is used in various gameplay situations to enhance realism (e.g., during walking, jumping, shooting) or to inform the player about a specific in-game event (e.g., taking damage). The essence of this effect is that the camera "shakes," shifting relative to its initial position with a set intensity for a specific duration. This effect is especially common in console games, where camera vibration is accompanied by controller vibration. A failure would be the absence of this effect in the situations described above.

Types of Hardware Failures in Video Game Graphics

- Screen tearing

© International Software Testing Qualifications Board

- Ghosting
- Flickering
- Level of detail (LoD) failures
- Z-Fighting
- Missing textures
- Visual artifacts (graphic "glitches")
- Failures in visual effects
- Lighting, reflections, and shadows failures

Screen Tearing

This failure is not directly related to the graphic objects of the game but arises due to hardware characteristics. However, it has a noticeable visual manifestation in the form of scene distortions. The failure occurs because monitors have a specific refresh rate (e.g., 60 Hz, 144 Hz, 165 Hz, 240 Hz), which determines the number of image updates per second. If the graphics card generates more frames than the monitor can display, frames start overlapping, creating a "tearing" effect.

Ghosting

It is a failure, which appears as the overlay of several displaced images of an object on the screen. It occurs when the previous image remains visible simultaneously with the current one during the transition to the next frame, creating a blurred or "ghosting" effect.

The main causes of this phenomenon are related to the screen refresh rate and response time. The refresh rate determines how often the monitor updates the displayed image, while the response time is the period during which the processor transmits a new image to the display, replacing the old one. If the response time is too short, the previous image does not fully disappear, leading to the occurrence of the ghosting.

Other causes of ghosting include:

- Temporal anti-aliasing algorithms, which rely on data from previous frames and, when improperly tuned or when motion vectors are insufficient, lead to the accumulation of trailing artifacts.
- Upscaling and temporal upscalers, where motion reconstruction failures or mismatches between motion vectors and actual geometry result in residual artifacts.
- Low or unstable frame rates, which amplify frame-to-frame differences and cause temporal filters to behave more aggressively.
- Motion vector defects (missing, inaccurate, or mismatched with animation, skinning, or VFX), causing the engine to incorrectly “predict” pixel movement.
- Post-processing effects (motion blur, bloom, volumetric effects), which, when combined with temporal filtering, further intensify trailing and ghosting artifacts.

Flickering

This often refers to a situation where an object (such as a texture or particles) periodically blinks, flickers, disappears, and reappears. Such behavior may be related to response time settings.

Level of Detail (LoD) Failures

LoD models are divided into levels (for example, LoD0 — the most detailed model, LoD1 — a slightly simplified one, and so on up to LoD3 or LoD4). The game dynamically selects the appropriate model level depending on:

- the distance of the object from the camera;

© International Software Testing Qualifications Board

- the current frame rate (FPS);
- the speed of object movement;
- the number of simultaneously visible objects in the scene.

Despite the effectiveness of the technology, its implementation may be accompanied by various failures. For example:

Abrupt "level popping" occurs when the transition between levels of detail is too noticeable.

On low-performance systems, the high-detail model may not load in time, and the player observes a rough, simplified version of the object even at close range.

Simplified LoD models may sometimes fail to preserve the correct shape or proportions of the original, which can affect the scene's aesthetics or confuse the player.

Z-Fighting

This rendering failure occurs when two primitives (basic geometric shapes used as a starting building block for creating more complex 3D models) alternately overlap each other, creating a flickering effect on the screen.

The reason for this failure is that both primitives are positioned so closely to the camera that the graphics processor cannot determine which one should be displayed in a specific pixel. This happens due to uncertainty in the operation of the Z-buffer, which manages depth, as the coordinates of the primitives nearly coincide.

When the scene changes, the failure manifests as flickering or graphic noise, as data from one of the primitives is alternately used to render the screen pixels.

Missing Textures

This failure appears as placeholders in place of missing textures. The game's code contains information about textures and their locations, but the textures themselves are either absent or the graphics processor fails to load them from storage in time to display them on the screen. [Gregory18]

Visual Artifacts (Graphic "Glitches")

These failures are difficult to classify within the previously described categories, and sometimes their exact cause is hard to determine. Such "glitches" are most often related to the unstable operation of the graphics processor, its drivers, or other hardware and software components.

An example of such failures is "wandering pixels", which are artifacts that appear around objects and disappear when the viewing angle changes.

Failures in Visual Effects

Failures in visual effects may include:

- Missing effects where they are necessary (e.g., no ripples on a pond's surface in windy weather).
- Violation of visual effect implementation principles, including realism (e.g., a grenade explosion consisting only of smoke).
- Excessive or insufficient speed of effect display (e.g., flames in a fireplace moving too fast).
- Effect duration being too short or too long (e.g., a gunshot muzzle flash persisting after the shot has been fired).
- Breaking the laws of physics and optics and failures in effect implementation (e.g., extremely dense fog completely obscuring the game scene).

© International Software Testing Qualifications Board

It is evident that testers, not being specialists in visual effect creation, cannot fully evaluate them. However, they can assess effects based on the principles of their implementation, identify where an effect is missing but should be present, detect its improper placement or timing, notice distortions, overly short or long durations, and unrealistic environmental interactions. Additionally, a tester can observe if an effect becomes repetitive and starts to annoy the player over time. These are the key aspects that should be identified during game graphics testing of visual effects.

Lighting, Reflections, and Shadows Failures

Proper lighting setup in video games, just like in cinema or visual arts, serves as an essential tool for achieving specific objectives: creating atmosphere, emphasizing object volumes, and simplifying navigation through the game level by harmoniously combining light and color.

Lighting helps highlight key points, such as save points or resource refill stations, indicate loot on the map, or mark cover spots where the player can remain undetected.

Lighting can be natural or artificial, directional, reflected, or refracted, and can also vary in intensity. Common lighting failures are typically caused by improper use of light sources, incorrect simulation of optical effects, or shadows cast by objects.

Lighting failures also include disrupted auto-exposure that is, the imitation of how human vision adapts when moving from darkness into a lit environment.

Another lighting failure is the improper use or handling of glare. Glare is used to add realism to a scene's lighting and to imitate very bright light sources. [Lee16], [Tavakkoli18], [Romero19]

4.2 Test Types in Video Game Graphics Testing

The process of creating graphic content, including two-dimensional or three-dimensional models, visual effects (VFX), implementation of lighting for game scenes, reflection systems, and shadows, is highly complex and consists of several key phases. It is essential to remember that testing should begin as early as possible within the software development lifecycle. Essentially, there are three test types for testing video game:

- Artistic Testing (verification of visual, including aesthetic aspects)
- Technical Testing (verification of technical parameters and requirements)
- Playtesting (includes verification of animation and object interactions)

It is important to understand that these are not separate independent activities within the graphics test process. The distinction between these test types is primarily aimed at more accurately identifying the root causes of defects and the specialists responsible for fixing them in the future. Additionally, each test type employs its own test techniques and test approaches, which differentiate test activities from one another. Different specialists may be involved at each phase, ranging from artists to regular users.

4.2.1 Video Game Graphics Artistic Testing

During the creation process, a graphic object undergoes numerous phases and tests. In large projects, specialists involved in developing graphic assets are often included in the test process.

Testing at this level is called artistic testing. The specialist's task is to evaluate the visual and aesthetic aspects of a graphic object, specifically:

- Object geometry
- Texture quality
- Compliance of models with their real (historical) prototype
- Model animation

© International Software Testing Qualifications Board

- Visual effects
- Scene lighting, object reflections, shadows
- Consistency of visual components (determining that textures, models, and effects appear as part of a cohesive world)

The goal is to enhance the visual quality of models before exporting them to the game engine. Extensive experience and access to development tools allow artists, 3D modelers, and animators to identify defects much faster and more efficiently. While art specialists can conduct artistic testing with greater efficiency due to domain expertise, game testers are also responsible for artistic testing, particularly for evaluating compliance with visual specifications, style guides, and reference materials provided in the test basis.

Artistic testing is a complex task that can be performed at various production phases, from creating basic geometry and textures to exporting the model to the engine and placing it on the map. For artistic testing, testers often do not require specialized tools or graphic content editors—it can be conducted directly within the game.

4.2.2 Video Game Graphics Technical Testing

The creation of graphics involves compliance with a large number of technical requirements, ranging from the polygon count of a 3D model to complex parameters for calculating object reflections. Technical testing is aimed at detecting defects related to:

- Graphic file formats
- Presence of necessary graphic files in the appropriate directories
- Integrity of these files
- Collision model size
- Correct placement of hitboxes
- Compliance of the model with the game engine's requirements regarding polygon count limitations
- Distance settings for model switching
- Verification of timely switching between destruction models and other technical aspects

The goal of technical testing is to verify that all technical parameters of graphic objects are met for their correct display in game scenes while also optimizing hardware performance on the gaming platform.

Graphics testing occurs in parallel with artistic testing and may require additional tools. Some tests can be conducted directly within the game engine or in a 3D editor.

4.2.3 Video Game Graphics Playtesting

At the gameplay level, the tester’s task is to verify the interaction of graphic objects in the game scene concerning their impact on gameplay. This may involve evaluating whether a model’s animation aligns with the in-game situation or testing whether game object settings comply with the requirements of the game mode.

It is fair to say that playtesting primarily identifies defects affecting the overall player experience. In other words, it determines how well the graphic objectss of a video game enhance immersion without disrupting gameplay.

The main failures detected at the gameplay level include:

- Desynchronization of animations with game events
- Incorrect rendering of effects during rapid camera movement
- Unnatural interaction of visual effects with the environment

© International Software Testing Qualifications Board

- Inconsistency of graphics with the game's artistic style
- Graphic failures when switching between different quality settings (low, medium, high)
- Lack of emphasis on important game components, causing them to blend into the background
- Excessive graphic objects that distract from the gameplay
- Flickering textures or "artifacts"
- Objects disappearing from view at certain camera angles
- Annoying or overly intrusive visual effects
- Excessive hardware load on the gaming platform

4.3 Executing Video Game Graphics Testing

The approach to testing video game graphic content does not involve radically different test techniques from those used in software testing. Since a graphic object is a system component, it is logically justified to apply a methodology based on verifying such components at different phases of their creation and integration. Consistent with the ISTQB Foundation Level test levels, graphics testing is organized into component (unit), integration, and system testing phases, each with distinct objectives and scope for graphics content. The primary difference in testing lies in the level of isolation of the graphic subsystem object. In this case, testing is performed sequentially as follows:

- Unit Testing
- Component Integration Testing
- System Testing (often including certification/compliance testing)

At the unit test level, objects are examined during their creation. Here, graphic objects undergo several key verifications:

1. Model Testing

- Geometry Testing – verifies that 3D models do not contain overlapping polygons or unnecessary components.
- Polygon Optimization – evaluates whether the model meets the specified polygon limits for a particular game engine.
- UV Mapping Testing – evaluates the correctness of the unwrapping for subsequent texture application.

2. Texture Testing

- Texture Quality – verifies the absence of artifacts, blurriness, or incorrect resolution.
- Compatibility – verifies that textures are correctly displayed in the game engine.
- Seams and Joints – evaluates that textures align smoothly without visible seams.

3. Testing of Real (Historical) Prototypes (where applicable)

4. Animation Testing

- Smoothness of movements – verifies the absence of jerky movements, pose inconsistencies, or other visual failures.
- Animation blending – verifies that transitions between animations appear natural.

5. Visual Effects (VFX) Testing

- Effect Physics Compliance – verifies whether the effect adheres to the laws of physics (or the game's rules).
- Effect Duration – verifies that the effect duration is appropriate.

© International Software Testing Qualifications Board

It is important to remember that testers may encounter certain challenges in this phase, such as:

- Some failures may only appear in motion or at specific viewing angles.
- Visualization results may vary depending on the graphics card or display.
- The test process often remains subjective and requires an expert approach.

Therefore, some tests for graphic objects, their animations, visual effects should be repeated later.

After a graphic object is created and tested, it is imported and tested within a test scene in the game engine to verify if it functions correctly in the game's context.

The primary tests include:

1. Collision Testing:

- The model interacts correctly with other objects in the game.
- The model has properly configured hitboxes.
- Interactions between objects and object behavior adhere to the laws of physics (or the game's rules).

2. Resource Optimization:

- There are no clipping failures.
- Level of Detail switching is properly configured.
- Lighting, reflections, and shadows are correctly set up and function as intended.
- There are no objects that create significant strain on the hardware of the gaming platform.

3. Visual Effects Testing:

- Visual effects do not overload the game scene or distract from the gameplay.
- A special effect is synchronized with specific actions and events in the game scene.

The final phase of video game graphics testing is comprehensive system testing, which often includes certification testing (for console games or after localization).

As part of system testing, the integration of the graphics subsystem with other video game subsystems is verified, along with compliance with all artistic and technical requirements. In this phase, key non-functional testing is also conducted, including performance efficiency testing, compatibility testing, and load testing.

Localization testing is a separate test type and will be described in detail later. However, for completeness, it should be mentioned that graphic content is one of the primary priorities when testing for legal, ethical, religious, and other regulatory compliance required for video games. Additionally, game platform developers impose specific graphic content requirements, and failure to comply may prevent a game from being published on a distribution platform.

© International Software Testing Qualifications Board

5 Audio Testing - 120 minutes

Keywords

Audio testing, content-audio testing, audio technical testing

Video Game Specific Keywords

Ambient, distortion, sound effect, sound zone, volume

Learning Objectives for Chapter 5

5.1 Principles and Concept of Audio Content in Video Games

GaMe-5.1.1 (K1) Recall the video game audio functions

GaMe-5.1.2 (K1) Recall the video game audio types

GaMe-5.1.3 (K1) Recall the video game audio techniques

GaMe-5.1.4 (K2) Give examples of failures in video game audio content

5.2 Test Types in Audio Testing

GaMe-5.2.1 (K2) Summarize content-audio testing.

GaMe-5.2.2 (K2) Summarize audio technical testing.

5.3 Executing Video Game Audio Testing

GaMe-5.3.1 (K3) Apply audio testing

5.1 Principles and Concepts of Audio Content in Video Games

5.1.1 Features of Video Game Audio

Video game audio is a crucial component that influences immersion, perception, and overall player experience. In modern video games, sound design serves as a storytelling, emotional, and interactive tool.

The primary objectives of game audio include supporting gameplay mechanics, creating atmosphere and emotional depth, providing information and player feedback. It is important to recognize that sound design is not just for entertainment but also serves several essential functions:

1. Signals. Certain sounds inform the player about resource shortages, enemy base captures, successful shots, and other important events.
2. Spatial orientation. Sounds help players determine whether they are in a closed space or an open environment.
3. Environmental awareness. Players navigate the game world using sounds associated with different surfaces, such as water, sand, or snow.
4. Notifications. Sounds indicate events that are not visually displayed on the screen, such as leveling up or receiving a new quest.
5. Atmospheric immersion. Ambient sounds enhance the feeling of presence, such as city noise, farm sounds, or battlefield chaos.

© International Software Testing Qualifications Board

6. Effect amplification. Sounds emphasize critical game events. For example, after a grenade explosion, the player may hear a ringing or muffled sounds, simulating a concussion effect.
7. Emotions and mood. Music and sound effects evoke emotions and set the mood. For instance, combat scenes feature intense and dynamic music, while exploration is accompanied by calmer melodies.

The same sounds can serve multiple functions. For example, alert or notification sounds can also contribute to the game's atmosphere.

5.1.2 Video Game Audio Types

Game audio can be broadly categorized into six types:

- Background music
- Graphic user interface sounds
- Ambient sounds
- Character sounds
- Sound effects
- Character voiceover

Background Music

Music is a fundamental part of the game and acts as its signature component. It sets and conveys the desired mood and can directly influence player actions—either motivating dynamic play or slowing down the pace.

Graphic User Interface Sounds

UI sounds play a functional role, providing player feedback. They confirm that an action or command has been successfully registered and processed by the game system (subsystem).

Ambient Sound

Ambient sounds reflect the characteristics of a specific location. These sounds usually play in the background and are independent of character actions. Examples include rustling leaves in a forest, ocean waves, city noise, or conversations of passersby. Such sounds immerse the player in the game world's atmosphere.

Character Sound

Character sounds include all noises made by characters except for speech. This may include footsteps, creaking doors, jingling keys, and other effects that enhance realism.

Sound Effects

Sound effects (SFX) serve various purposes, from adding realism to creating unique audio cues, such as the sounds of magical spells or alien technology.

Character Voiceover

Voiceover is used for both playable and non-playable characters, adding variety and depth to the soundscape. It can be modified with additional effects, such as voice distortion or simulated poor reception. However, the absence of voice acting does not always negatively impact the game's atmosphere.

© International Software Testing Qualifications Board

5.1.3 Video Game Audio Techniques

To create realistic and immersive audio, it is necessary to consider the spatial distribution of sound and the human auditory system's perception capabilities. For this, the following techniques are used:

Sound Zones

Sound zones are segments of the game environment where the set of sounds, their intensity, volume, and characteristics differ from the surrounding areas. Thus, sound zones help players form a clear sense of space.

Binaural Sound

The binaural sound effect is based on the fact that humans hear with both ears simultaneously. When a person turns their head, sound reaches one ear before the other, allowing them to determine the direction of the sound source and even estimate its distance.

Spatial Propagation

In the real world, sounds never travel unchanged—they are reflected, absorbed, or distorted depending on obstacles. If a game fails to adjust sound properties based on whether the source is behind a wall or in an open field, it breaks immersion and disrupts the atmosphere.

To create a realistic soundscape, games must account for various sound propagation properties and their effects, such as occlusion, reverberation, and more.

5.1.4 Types of Audio Failures

Despite the wide variety of audio components in video games, failures characteristic of this subsystem can be grouped into the following categories:

- Missing sound or sound effect,
- Playback of incorrect sound,
- Incorrect playback of the correct sound,
- Sound control failures (inability to adjust character voice volume, music level).

Missing sound or sound effect

The sound does not play even though it should for a specific character action or at a particular moment.

Playback of incorrect sound

Such failures may include situations where:

- playback of a sound that does not match the game situation;
- playback of a sound that does not correspond to its real (historical) prototype.

Incorrect playback of the correct sound

These failures are the most numerous in their variety. They include:

- Interruptions and cutoffs: sudden disappearance of sound or abrupt volume jumps.
- Looping: a short sound gets stuck in infinite repetition.
- Distortion: sounds play in a distorted and unintelligible manner.
- Desynchronization: the sound lags behind or leads the displayed animation, object, or game situation.
- Incorrect volume: the playback volume is either excessively high or too low.

© International Software Testing Qualifications Board

- Alteration of sound properties: additional effects such as binaural sound effect, reverberation, occlusion, echo are missing.
- Sound zone configuration: sounds from one zone are heard in another.
- Noise, crackling: sound is overlapped by extraneous noise, crackling, hissing, clicking.
- Noticeable “seams” between background compositions: an overly abrupt transition between background compositions.

It is important to understand that some of the situations described above may be intentionally created according to the video game’s storyline (e.g., simulating poor communication quality, impaired character hearing after a concussion). Therefore, when identifying a defect, it is crucial to consider the context in which a particular sound is used.

Sound control failures

Audio files may be of perfect quality and located in the correct directories, but the player may be unable to:

- increase or decrease character voice volume,
- adjust the background music level,
- mute the narrator’s voice, despite such functionality being declared in the video game and available in the corresponding section of the user interface.

Such failures may also be identified during UI testing.

5.2 Test Types in Video Game Audio Testing

The audio failure types described in the previous sections stem from two categories of underlying defects: defects in audio content (wrong, missing, or incorrectly triggered assets) and defects in the audio technical infrastructure (faults in rendering, mixing, or platform integration). Accordingly, audio testing is divided into two complementary types, each targeting one defect category:

- Content-Audio Testing
- Audio Technical Testing

5.2.1 Video Game Content-Audio Testing

Sound in games is a holistic system of components: character voices, musical compositions, environmental effects, UI sounds, footsteps, wind noise, engine sounds, distant battle echoes, or rustling leaves. All these components must be well-balanced and structured. If the soundscape is chaotic and unorganized, the player may feel discomfort, overload, and confusion. On the other hand, if key sounds (e.g., enemy footsteps, item pickups, important indicators) are too quiet or unclear, the player may miss crucial information, negatively impacting gameplay success.

As part of content-audio testing, tests are conducted to identify defects in audio quality, as mentioned earlier. These tests focus not only on the presence of audio cues but also on their clarity, directionality, volume balance, and timing. The way sounds interact with visual cues and gameplay logic is also closely evaluated. For instance, if a warning sound doesn’t coincide with an incoming threat on-screen, it can break immersion or cause frustration.

Moreover, testers examine how audio assets respond to dynamic in-game changes, such as weather transitions, day-night cycles, or player actions. Environmental reverb, occlusion, and other effects are analyzed for spatial realism.

© International Software Testing Qualifications Board

Additionally, localization testing evaluates whether translated voiceovers sync properly and preserve emotional tone. Glitches like looping defects, crackling, or missing audio are flagged during playthroughs. Overall, effective content-audio testing evaluates if the auditory environment supports the game's narrative, gameplay clarity, and emotional depth—turning a functional world into a believable, reactive one.

5.2.2 Video Game Audio Technical Testing

Video game audio technical testing refers to the systematic process of evaluating and verifying different qualities of audio components within a game. As video games evolve into more immersive experiences, the importance of rigorous audio technical testing cannot be overstated. Audio technical testing goes beyond simple playback checks. It covers:

- Functional suitability: Verifying that audio triggers correctly and plays as intended during gameplay. This includes testing scripted and emergent events to evaluate if audio is contextually appropriate and responsive.
- Consistency: Evaluating that volume levels, transitions, and effects are coherent throughout different environments and hardware setups. Testing must account for audio layering, fade-ins and fade-outs, and crossfading between music tracks or ambient loops.
- Performance: Measuring resource usage to prevent lag or audio glitches, especially in high-demand scenarios. Audio load balancing and priority systems must be evaluated under stress conditions.
- Platform Compatibility: Testing across various devices (PCs, consoles, mobile) to verify reliable audio performance. Different drivers, APIs, and sound hardware configurations can significantly impact output and must be accounted for.

These test types are also used to detect failures related to the presence of necessary audio files in the system, their correct usage in different situations, and the player's ability to control game audio within the advertised functionality (turning sounds on and off, adjusting volume levels, selecting audio languages, or switching between surround and stereo modes). Testers must also evaluate accessibility settings, verify lip-sync with dialogue, and evaluate how audio interacts with gameplay mechanics like stealth, cutscenes, or cinematic sequences.

5.3 Executing Video Game Audio Testing

Content-audio testing and audio technical testing are applied together during test activities, just as with other video game subsystems.

The tester must conduct both test types to obtain a complete understanding of how correctly the audio is used and configured. The number of tests depends on which objects need to be tested.

Audio testing often begins at later phase of development. This is because the core game content (object/character models, maps, items) is typically created first, while the audio components are added later. By the time audio testing begins, the object itself usually has a completed model with animation, integrated into the client, but may not yet have sound.

Audio samples should first be pre-tested by the specialists who created them (e.g., audio engineers, Foley artists, and sound effect designers). These specialists may participate in testing individual sounds before submitting them for further integration into the client.

In the content-audio testing phase, which is the most extensive phase of audio testing, the tester verifies the following:

- Presence of sound for specific game situations.

© International Software Testing Qualifications Board

- Relevance of sounds to the object and game scenario.
- Absence of extraneous noise or distortion in game event sounds.
- Avoidance of irritating sounds.
- Appropriate audio volume levels.
- Correct positioning of sound based on its source.
- Proper sound zone configuration.
- Character voice acting matches the game context and audience expectations.
- Sounds correspond to real-world analogs (historical accuracy).
- Overall sound balance, including musical accompaniment.
- Failures such as looping, distortion, missing sounds, delays, or cut-offs.
- Lack of effects that provide a realistic audio environment.

The tester must perform all of the tests listed above to develop a comprehensive assessment of the correctness of audio configuration and usage. The number of tests conducted depends on which specific objects require verification.

In addition to content-audio testing, the specialist must verify the following:

- The game allows the enabling and disabling of sounds and/or music.
- Volume levels can be adjusted.
- All audio files have consistent naming conventions and are stored in the correct directories within the project.
- Audio sources and zones are correctly placed on the game map.
- Sounds play correctly on various audio systems (e.g., headphones, speakers, studio monitors, surround systems, home theaters, and other audio formats and technologies).

The final phase may include performance testing, which evaluates how audio affects the hardware resources of the gaming platform.

When assessing audio environment quality, the tester must consider the following criteria:

- The audio does not cause FPS drops in a sound-intensive scene.
- The audio does not lead to memory leaks.
- The set limits on the number of simultaneous sounds (e.g., max concurrent sound effects) are adhered to.

If an immersive audio environment significantly impacts performance, a lightweight audio option is available for less powerful devices.

© International Software Testing Qualifications Board

6 AI and Physics Testing - 180 minutes

Keywords

none

Video Game Specific Keywords

Pathfinding, rigid body physics, soft body physics, video game physics

Learning Objectives for Chapter 6

6.1 Principle and Concepts of Artificial Intelligence (AI) and Physics in Video Games

GaMe-6.1.1 (K2) Distinguish the main aspects of video game AI

GaMe-6.1.2 (K2) Give examples of failures of video game AI

GaMe-6.1.3 (K2) Distinguish the main aspects of video game physics

GaMe-6.1.4 (K2) Give examples of failures of video game physics

6.2 Executing AI and Video Game Physics Testing

GaMe-6.2.1 (K3) Apply AI testing

GaMe-6.2.2 (K3) Apply video game physics testing

6.1 Principles and Concepts of Artificial Intelligence (AI) and Physics in Video Games

6.1.1 Video Game Artificial Intelligence Features

Artificial Intelligence (AI) in video games refers to computational systems that create autonomous, responsive, and adaptive behaviors in game entities and game systems. While AI is most commonly associated with non-player character (NPC) behavior—such as enemies, allies, and neutral characters — it also encompasses broader game systems including:

- Dynamic difficulty adjustment
- Procedural content generation
- Intelligent camera systems
- Dynamic and adaptive environments
- Strategic decision-making

In multiplayer-focused games, AI-controlled characters that substitute for human players are specifically referred to as "bots".

The core AI algorithm follows a sequence of steps: "sense → think → act." Developers can add complexity to each phase, making NPCs appear more intelligent.

© International Software Testing Qualifications Board

To better understand the root causes of AI-related failures, it is important to know which architectures are used to implement AI in a particular game. Common AI architectures include:

- Finite State Machine (FSM) – State-based behavior with discrete transitions
- Behavior Tree (BT) – Hierarchical, modular decision structures
- Utility AI – Score-based action selection using evaluation functions
- Goal-Oriented Action Planning (GOAP) – Dynamic planning to achieve goals
- Blackboard Systems – Shared knowledge repositories for multi-agent coordination
- Hierarchical architectures – Layered systems combining multiple approaches.

Using these methods, hundreds of great games have been developed, where virtual opponents closely resemble real players. Understanding these methods helps identify weaknesses and detect defects in different situations.

However, game AI is always based on predefined parameters set by developers, making its behavior predetermined and limited by the creator's design. This is a key drawback of AI architectures. Additionally, creating believable and intelligent NPC behavior requires significant development time to account for numerous scenarios and potential reactions. Due to this complexity, defects frequently occur in AI behavior.

Furthermore, NPCs are usually restricted in their movement paths within game levels. To define where an NPC can navigate, developers use navigation meshes (NavMeshes). This is a data structure that describes the game world's surfaces, enabling NPCs to calculate paths from one point to another (Pathfinding).

6.1.2 Types of AI Failures in Video Games

AI failures in video games can significantly impact gameplay quality, immersion, and player satisfaction. These failures generally fall into the following categories:

- Character interaction failures
- Movement and pathfinding failures
- Perception and awareness failures
- Decision-making and behavioral logic failures
- Group and swarm behavior failures
- Performance-related failures
- Realism and game design balance failures

Character Interaction Failures

- Failures in how AI-controlled characters interact with the player, environment, or other NPCs.
- NPCs ignore player commands or presence.
- Allies fail to support or react appropriately.
- Characters do not use environmental components such as cover.
- Characters behave as if unaware of environmental barriers (e.g., "see" through walls).

Movement and Pathfinding Failures

- Failures related to navigation and motion logic.
- Characters become stuck in place or walk into obstacles.
- Pathfinding defects cause navigation through inaccessible areas.
- Characters loop endlessly in action sequences (AI stuck in loop).
- Incorrect routes are selected due to defective navigation meshes (NavMesh).

© International Software Testing Qualifications Board

Perception and Awareness Failures

- Breakdowns in sensory simulation and situational responsiveness.
- Enemies fail to detect or respond to the player.
- Characters forget about the player too quickly in stealth mode.
- NPCs lack awareness of events in their surroundings.

Decision-Making and Behavioral Logic Failures

- Failures in action prioritization or strategic responses.
- AI prioritizes incorrect or illogical actions.
- Characters behave irrationally or inconsistently in similar situations.
- Enemies attack the wrong targets (e.g., allies instead of the player).

Group and Swarm Behavior Failures

- Breakdowns in coordinated multi-agent behavior.
- Crowd movements appear chaotic or unnatural.
- AI characters in groups fail to interact with one another logically.

Performance-Related Failures

- AI behavior causes system performance degradation.
- Frame rate (FPS) drops due to complex or unoptimized AI logic.
- AI freezes or stalls in high-load or unexpected situations.

Realism and Game Design Balance Failures

- Confusion between defects and intentional design choices.
- AI behaves too predictably or unrealistically.
- Characters violate expected physical laws (e.g., float, pass through objects).
- Overly intelligent AI frustrates players, while overly passive AI may reduce challenge.

However, it is important to consider "game design conventions". Absolute realism in character behavior can hinder game design and detract from player enjoyment.

6.1.3 Video Game Physics Features

The adherence to the laws of physics is unquestionable for most players in almost any game. Game physics serves multiple purposes, but its primary goal is to make the game intuitive and engaging. When objects behave unpredictably, players struggle to understand the rules of the game and may lose immersion or experience gameplay frustration.

At the same time, as with AI, it is important to remember that the video game world has its own rules, and absolute realism can sometimes worsen the gameplay experience if all physical laws are strictly followed. Overly accurate physics simulations may hinder game flow, reduce responsiveness, or even make gameplay boring or repetitive, especially in arcade or action-heavy genres. It is crucial to remember that the main purpose of a game is entertainment, not teaching physics fundamentals. Therefore, when discussing physics failures in video games, only clear violations of key physical laws should be considered.

If a game is set in a non-fantasy world, such failures are usually related to the absence of fundamental forces acting on an in-game object, such as:

- Gravity (defect: an object levitates or shoots upward to an extreme height);
- Inertia (defect: a fast-moving heavy object stops instantly upon abrupt deceleration);

© International Software Testing Qualifications Board

- Friction (defect: a heavy object slides for too long on a non-slippery surface);
- Centrifugal force (defect: a car takes a high-speed turn on the outer lane and successfully completes it without braking);
- or cases where objects exhibit properties impossible in the real world, such as extreme speeds, unrealistic elasticity, or unbreakable surfaces under massive impacts.

In video games, physics is generally divided into two main categories:

Rigid Body Physics (RBP) and Soft Body Physics (SBP).

Rigid body physics refers to the simulation of physical behavior for objects that are assumed not to deform when forces are applied. In a game context, rigid bodies maintain a fixed shape and size while reacting to gravity, collisions, forces, impulses, and constraints according to predefined physical rules.

Soft body physics describes forces that affect an object's shape, such as the movement of flags, water, jelly-like materials, or character clothing—essentially, any object that deforms under external forces. These effects significantly enhance realism and immersion, especially in slow-motion or cinematic sequences.

The main challenge in implementing soft body physics is the massive number of calculations required. Achieving realistic effects demands millions of operations per second, often exceeding processor capabilities. As a result, soft body physics is frequently simplified. For non-essential objects, looped animations or procedural effects are often used instead of real-time physics calculations. Such simplifications are not considered failures as long as the animation appears natural and is consistent with the game's visual style and expectations.

6.1.4 Types of Game Physics Failures

When discussing game physics, it primarily refers to rigid body physics.

All failures related to violations of physical laws in games can be broadly classified into two groups by defects that caused them:

- Destruction defects
- Object behavior defects

Failures caused by destruction defects include:

- Destruction without clear cause or impact response. Objects break apart without a visible reason or fail to react correctly to player actions, environmental forces, impact direction, or impact strength.
- Unrealistic or repetitive destruction. Objects break apart unnaturally or identically regardless of conditions.
- Delayed or missing destruction. Destruction occurs with a noticeable delay or does not occur when expected.
- Incorrect post-destruction collision. Destroyed objects or debris incorrectly block players, NPCs, vehicles, projectiles, or navigation.
- Incorrect disappearance or persistence of debris. Destroyed objects vanish too quickly, disappear without debris, or remain in an incorrect visual/physical state..

Failures caused by object behavior defects include:

- Unrealistic object movement. Objects fly, fall, bounce, slide, or roll at unnatural speeds, along incorrect trajectories, or without an apparent physical cause.

© International Software Testing Qualifications Board

- Lack of response to physical forces. Objects fail to respond to gravity, collisions, explosions, or other applied forces, for example by remaining suspended in the air or staying immobile after impact.
- Incorrect response to collisions and explosions. Objects fail to react appropriately to impacts or blast waves: they may not break, deform, move, or show visible damage, or may do so without considering force, direction, or speed.
- Interaction without physical contact ("telekinesis"). Objects affect, push, move, or damage each other without actual contact or a valid collision event.
- Collision tunneling. A fast-moving object, such as a bullet, shell, arrow, vehicle, or physics object, passes through another object because the collision is not registered by the system.

Physics in video games is one of the most complex aspects of development. The challenge for developers is to find a balance between hardware performance limitations and realism. At the same time, it is essential to remember that a game should be engaging for players above all else. This influences the design approach to game physics — in some cases, realism must be sacrificed in favor of interesting gameplay mechanics and an improved gaming experience.

6.2 Executing AI and Video Game Physics Testing

6.2.1 Executing AI Testing in Video Games

When planning AI testing, it is necessary to understand what exactly needs to be verified. The test objectives may include the verification of:

- Functional correctness: AI must perform its tasks correctly (e.g., enemies should react to player actions, and NPCs should interact with the environment, objects, or other characters in a meaningful and consistent way).
- Behavioral diversity: Characters and objects should exhibit natural and varied behavior, avoiding predictability. This includes non-repetitive pathfinding, decision-making influenced by changing context, and reactions that adapt to the player's choices.
- Balance: AI should not be too difficult or too easy, providing engaging gameplay across various skill levels. AI behavior must align with player progression and be appropriate for different game modes or difficulty settings.
- Performance: AI should operate efficiently without significantly increasing system load. Poorly optimized AI can lead to dropped frames, lag, or unresponsive game logic.

At the beginning of AI testing, it is essential to verify basic functions that AI should perform or control. For example:

- Interaction with the environment: AI should correctly navigate obstacles, use cover, and interact with objects.
- Logical sequences: AI should correctly execute a sequence of actions (e.g., gathering resources, building shelters, patrolling).
- Reaction to triggers: For example, an enemy should attack the player upon spotting them or hide when health is low.

This activity is often automated using test scripts or built-in tools in game engines to reduce the time spent on repetitive tests.

AI must exhibit realistic behavior. To determine this, the following aspects are tested:

- Variety of actions in similar situations.
- Randomness and variability in decision-making to prevent complete predictability.
- Natural interactions with the player and the environment.

© International Software Testing Qualifications Board

AI should be well-balanced to keep the game engaging. For example:

- Enemies should be intelligent enough to pose a challenge but not so difficult that defeating them becomes impossible. They should also exhibit weaknesses the player can learn and exploit.
- Non-player characters (NPCs) should provide useful opportunities for the player without being overly "perfect" or monotonous, enhancing immersion through subtle, believable actions and dialog.

During this activity, difficulty parameters are adjusted, and the overall gameplay experience is analyzed through playtesting, with real player feedback guiding further AI tuning.

AI can significantly impact CPU and memory performance, especially in games with a large number of NPCs. Performance testing is conducted using stress tests, where a large number of AI actors are launched simultaneously, performing diverse tasks. The impact of AI processing on FPS, frame time, and resource usage is measured, providing game modifiability under high-load conditions.

6.2.2 Executing Video Game Physics Testing

The physics engine serves as the foundation for object interactions in a video game. Its performance directly affects the realism and credibility of the gameplay. Testing game physics is a complex process that requires detailed analysis and verification of numerous scenarios.

It is necessary to determine which aspects of physics should be tested. The key areas include:

- Correct object behavior: Verifying that the engine properly implements physical laws (e.g., gravity, friction, inertia).
- Realism: Verifying that object movements and interactions look natural and do not create dissonance for players.
- Performance: Evaluating the impact of the physics engine on the game's performance.

If the game's narrative does not specify an alternate physical reality, then basic functional tests may include:

- Gravity:
  - Objects are attracted to other larger objects.
  - Objects fall with natural acceleration.
- Collisions:
  - Objects correctly interact with each other and with surfaces.
  - Objects do not pass through each other.
- Friction and sliding:
  - Objects slide on different surfaces depending on their mass and material composition.
  - Objects slow down correctly on various surfaces.
- Other physical laws and properties:
  - Objects sink in water depending on their material and shape.
  - Objects experience air and water resistance.
  - Objects leave marks on different surfaces.

The realism of the game depends on whether various game objects obey the laws of physics. To evaluate if players can fully immerse themselves in realistic interactions, the following should be tested:

- Realistic destruction of objects.

© International Software Testing Qualifications Board

- Realistic interaction between objects.

Video game physics testing is conducted to identify the failures described above.

The physics engine can place a significant load on the processor, especially in games with a large number of objects or detailed simulations. Performance is tested using:

- Frame processing time measurements: Verifying how much time is required to calculate physics in each frame.
- Memory usage measurements: Verifying how much memory is needed for physics calculations per frame or scene.

These video game physics tests are performed using built-in game engine tools or specialized software.

Video game physics testing is a meticulous process aimed at achieving realism, performance, and modifiability. Only thorough testing evaluates if physics meets player expectations and becomes an integral part of an immersive gaming experience.

© International Software Testing Qualifications Board

7 Localization Testing - 145 minutes

Keywords

Compliance, internationalization, localization, root cause

Video Game Specific Keywords

Cultural adaptation

Learning Objectives for Chapter 7

7.1 Principles and Concepts of Video Game Localization

GaMe-7.1.1 (K1) Recall the localization core concept

GaMe-7.1.2 (K2) Summarize Features of video game localization

GaMe-7.1.3 (K2) Distinguish between full and partial localization

GaMe-7.1.4 (K1) Recall root causes of localization defects

GaMe-7.1.5 (K2) Give examples of failures of video game localization

7.2 Approaches to Localization Testing

GaMe-7.2.1 (K2) Summarize linguistic testing and adaptation testing

GaMe-7.2.2 (K2) Summarize localization technical testing

7.3 Executing Localization Testing

GaMe-7.3.1 (K3) Apply localization testing.

7.1 Principles and Concepts of Video Game Localization

7.1.1 Video Game Localization Essentials

It is essential to note that localization is not just a simple translation into multiple languages. The video game localization process includes [Retsker81]:

- Translating in-game dialogue texts, subtitles, tooltips, descriptions, messages, character names, and item names while considering regional specifics.
- Translating cutscenes, interface components, and menus.
- Redrawing textures and graphics.
- Casting voice actors, dubbing, and recording audio files.
- Integrating localized content into the game.
- Translating and adapting the game's website and printed materials.
- Translating marketing materials.
- Providing post-launch support for the localized version.

To mitigate risks of adapting a game for a particular region, developers use a process known as internationalization. This is the process of designing and developing a product in a way so that localization becomes easy with as little effort as possible. It usually consists of resource

© International Software Testing Qualifications Board

externalization, Unicode/encoding support, locale aware formatting, pluralization, text expansion considerations, bidirectional and right-to-left support, and flexible user interfaces/layouts. [Chandler11]

Internationalization includes:

- Designing and developing software in a way that does not create obstacles for localization.
- Extracting localized components from the code or content so that localized versions can be loaded later or selected based on user preferences
- Allowing the use of components that cannot be implemented before the localization process.
- Supporting regional, linguistic, or cultural references (including fonts and formatting).

This list does not necessarily include the localization of content, programs, or products. Instead, it outlines software development approaches that make future localization easier and more efficient.

7.1.2 Features of Video Game Localization

The difference between localization testing for video games and application software is based on the understanding that these two types of products differ significantly in several key aspects, as discussed in previous chapters.

Key Differences:

- Adaptation of graphic content
- Adaptation of audio content
- Adaptation of textual content

As a result, these video game subsystems require mandatory adaptation, considering factors such as historical accuracy, religious, ethical, cultural, political, and ideological aspects of the target audience.

Localization testers must verify that symbols, gestures, references, and dialogues are appropriate for the local context. In some cases, failure to meet these requirements may lead to ambiguous reception by the target audience or even result in the product being banned in certain countries or regions. For example, content depicting religious figures, politically sensitive flags, or controversial slogans may need to be censored or altered to comply with local regulations. [URL3]

Existing game development traditions, connections to literature, cinema, and other media, as well as the deep genre convergence of gaming products, require adherence to specific norms and standards. This includes preserving lore consistency, narrative tone, and thematic coherence in translated content. Violating these standards may cause negative reception among players from different regions and disrupt immersion.

For example, in massively multiplayer role-playing games (MMORPGs), localization must consider established conventions for player race names, item names, and other genre-specific terminology. Consistency with prior installments or related franchises is also essential, especially for fan-favorite series. Moreover, UI layout, font size, text wrapping, and suchlike are crucial components that require thorough testing to avoid mismatches and defects in the localized build.

7.1.3 Full and Partial Localization

The scope of localization testing depends on the type of localization used by the game developers. The completeness of localization is influenced by various factors, but this syllabus does not cover them in detail. The key takeaway is that the extent of localization testing directly depends on the level of localization previously performed for the video game.

The following types of localization can be identified:

© International Software Testing Qualifications Board

- Full localization / cultural adaptation
- "Box" localization
- Text localization
- Localization with voiceover
- Graphic localization

Full Localization / Cultural Adaptation

Cultural adaptation occurs when the original game is completely reworked during the localization process. Only the source code and mechanics remain unchanged, while textures, storyline, dialogues, and character models may be entirely redesigned, effectively creating a new version of the game using the same engine.

This method is rarely used, but it is necessary in cases where a game cannot be released and sold in a specific market without such adaptation.

"Box" Localization

If a game is released and sold on a physical medium, the packaging text must be localized.

If a game is sold digitally on an online platform, only the store page is translated, including the game description and screenshots. "Box" localization does not extend beyond this scope.

Text Localization

In most cases, all in-game text must be translated, including subtitles. This means that a player may hear dialogue in a foreign language but will see subtitles in their native language to understand the conversation.

Localization with Voiceover

In addition to text translation, character speech and dialogues are dubbed by voice actors in the target language.

Graphic Localization

Every video game contains graphic objects such as textures and visual assets that may include embedded text—for example, writing on a wall, signs, newspapers, or shop banners.

Graphic localization verifies that all such text within the game is translated. This may include:

- Newspapers
- Storefront signs
- Road signs

Since many games are set in specific locations, the localization approach may vary.

- If an in-game object (e.g., a newspaper article) is crucial to the story, it must be translated to preserve key narrative components.
- However, other text components, such as storefront signs, may remain untranslated if they do not significantly impact the player's experience or storyline progression.

7.1.4 Root Causes of Localization Defects

Even major companies are not immune to "cultural" defects in their products. In some cases, a single phrase or joke can lead to a complete ban of the game in certain countries. Such failures not only damage the game's reputation but can also lead to financial losses and public backlash.

© International Software Testing Qualifications Board

The main root causes in the localization process that can result in localization defects include:

- Insufficient qualifications of translators, especially those without experience in the gaming industry or lacking creative adaptation skills.
- Lack of knowledge about the history, culture, ethics, worldview, religion, and legislation of the region where the game is intended to be released, leading to misinterpretations or offensive content.
- Failure to understand the connections between in-game components, storylines, episodes, and references to pre-existing works in literature, cinema, music, or other video games, which can break immersion or cause inconsistencies.
- Insufficient internationalization support in the game's technical infrastructure, such as inappropriate text encoding, inadequate font rendering pipelines, or insufficient support for right-to-left scripting.
- Insufficient skills of the voice actors, especially in conveying tone, emotion, or character identity, which may distort the intended player experience and lead to player dissatisfaction or criticism in reviews.

7.1.5 Video Game Localization Failures

Considering the root causes of localization defects mentioned above, the following categories of failures can be identified:

- Failures caused by translation defects
- Failures related to content adaptation for regional requirements
- Convergence failures
- Failures related to technical aspects (including lack of internationalization support)
- Failures related to dubbing (voiceover)

Failures Caused by Translation Defects

- Lack of valid translation
- Incorrect grammar, spelling, and punctuation
- Incorrect number formatting, currency units, calendars, and dates
- Incorrect addresses and phone numbers
- Incorrect usage or conversion of measurement units or currencies
- Incorrect display of numerical information
- Incorrect proper names
- Mistranslation of historical events, holidays, cultural references, proverbs, idioms geographic names (toponyms), slang, and abbreviations
- Distortion or omission of special characters (e.g., incorrect rendering of diacritical marks in French, Hebrew, or Arabic)
- Translation that does not match the context of the game
- Translation that does not comply with platform manufacturer requirements
- Over-localization. Not everything should be localized—some components should retain their original form, such as trademarks, logos, abbreviations, and product names

Failures Related to Content Adaptation for Regional Requirements

- Failures caused by ignoring different cultural perceptions of humor, satire, and jokes, including the acceptability of humor regarding specific subjects.
- Failures due to misunderstanding the target audience’s views on religious topics, including religious objects, rituals, and ceremonies, as well as attitudes toward marginal cults (e.g., Satanism, paganism).

© International Software Testing Qualifications Board

- Failures in historical accuracy or failure to consider alternative interpretations of historical events by the target audience. This includes:
  - One-sided portrayals of historical events, military conflicts, and scientific discoveries.
  - Misrepresentation of existing objects, cultural heritage, or traditional lifestyles of specific nations.
- Failures caused by a misinterpretation of national culture and worldview. This includes ignoring:
  - National stereotypes
  - Traditional cuisine
  - Symbols and color meanings
  - Clothing styles
  - Lifestyles of different ethnic groups
  - Attitudes toward children, LGBTQ+ communities, and animal cruelty
- Failures related to legal restrictions, including compliance with:
  - Age ratings
  - Child protection laws
  - Religious regulations
  - Restrictions on violence, drugs, sexual content, race, terrorism, political criticism, and government opposition
  - Bans on specific symbols and imagery

Convergence Failures

- Loss of recognizability and identity of a literary, cinematic prototype, or other cultural phenomenon after localization.
- Inconsistencies in established terminology and naming conventions between the video game and its literary, cinematic, or other original references.
- Legal failures related to license violations and copyright infringement.

Failures Related to Technical Aspects (Including Lack of Internationalization Support)

- Corrupted Text / Garbled Characters. Corrupted text appears when decoding text with an incorrect character encoding, resulting in random symbol substitutions, often from a different writing system. This usually makes the text unreadable.
- Localized text exceeding interface constraints (cut-off or scrolling failures). A term translated from one language to another may have a different character length, leading to truncated text in UI components.
- Misalignment. After localization, UI layouts may require reconfiguration to preserve the original alignment.
- Missing line-by-line translation. Testers should evaluate dialogue box text, images, and screenshots in documents or the user interface to verify all content is properly localized according to user expectations.
- Overlapping components. Occurs when UI components overlap, making text or controls unreadable or inaccessible.
- Font color. Some text may be unreadable because its color matches the background color of the current scene.
- Missing text. Some text may be lost during translation or development.
- Incorrect font/size. Different countries use different default fonts and sizes, which may not affect functionality but can significantly impact user experience by making text difficult to read.
- Incorrect hotkeys. In some cases, a hotkey may be unavailable because the corresponding letter does not exist in the localized language or keyboard layout.
- Variables in text templates. Different language types (analytical, synthetic) use various declensions, cases, and plural forms that affect word structure. A common error among

© International Software Testing Qualifications Board

developers is splitting sentences containing variables, leading to incorrect grammar in the target language. Developers often fail to account for how variables change sentence structure in certain languages.
- Audio sequence desynchronization between original and translated content. Translated character phrases may be longer or shorter than the original due to differences in syllable count. Since cutscenes in games are often pre-recorded, localization teams must carefully balance translation accuracy with syllable count while providing lip-sync correctness with character animations.

Failures Related to Dubbing (Voiceover)

- Unnatural-sounding speech.
- Inappropriate voice casting for characters.
- Failure to account for national and individual speech characteristics of characters.

One of the most obvious consequences of incorrect localization is the deterioration of the user experience. Poor translation, grammatical failures, or cultural mismatches can lead to misunderstanding the storyline, character traits, and video game mechanics. Players may struggle with completing quests or navigating the interface, reducing their enjoyment and increasing frustration levels.

Incorrect cultural adaptation can result in misinterpretations or even offense to the target audience. This not only harms the perception of the game but may also lead to boycotts, lawsuits, or other legal consequences.

7.2 Test Types in Localization Testing

Localization testing involves verification of the content of a video game application for compliance with linguistic and cultural requirements, as well as the specifics of a particular country or region. These test types help identify localization defects or translation failures in the localized version before the final product reaches the user.

To optimize the localization test process, different test types are used, depending on the roles involved and the level of parallel execution. Specifically:

1. Linguistic Testing (translation, spelling, grammar) and Adaptation Testing (cultural, religious, legal adaptation).
2. Technical Testing, i.e. internationalization.

7.2.1 Linguistic Testing and Adaptation Testing

Linguistic testing involves a series of tests to verify that:

- Consistent terminology is used throughout the game.
- There are no grammatical errors.
- There are no spelling errors.
- Hyphenations are correct
- Punctuation rules are correctly followed.
- The correct text direction is applied (right-to-left or left-to-right).
- The translation is accurate and considers the risks and causes of localization failures described earlier.
- Correct names for trademarks, cities, locations, job titles are used.

At the same time, the tester’s task includes evaluating the quality of video game content to meet the target region or user group requirements.

© International Software Testing Qualifications Board

Critical areas for content appropriateness and compliance:

- Humor
- Religion, including occultism
- History
- Morality and ethics
- Prejudices and stereotypes related to culture and people
- Politics
- Regional cultural norms
- Superstitions and symbolism (including color and number symbolism)
- Attitudes toward sex and LGBTQ+ representation
- Attitudes toward children and the elderly
- Attitudes toward animals and pets
- Legal restrictions
- Connections between the video game and literary, cinematic, or other media sources

Given target-market expectations and applicable legal, regulatory, and platform requirements, localization testing shall treat culturally sensitive content as a high-risk area.

Within this test type, video and audio content must also be tested to verify:

- Whether the audio matches the game component it belongs to.
- Whether the audio corresponds to the character's gender.
- Sound clarity (absence of interference, proper synchronization of tone between the original and translated versions, and balanced volume levels).
- Authenticity of the audio, including historical accuracy.
- Stylistic and cultural aspects of the audio (accents, speech characteristics).

7.2.2 Localization Technical Testing

The objective of localization technical testing is to verify internationalization and technical aspects of localization.

Localization technical testing has the goal to evaluate all game components after the integration of localized content to verify that they function correctly. This test type considers the aspects including text display, interface functionality, audiovisual synchronization, subtitle correctness, audio file playback, and overall performance. If these aspects are overlooked, players may encounter failures that degrade their gaming experience, such as missing subtitles, broken UI layouts, incorrect audio triggering, or system crashes tied to specific language settings.

Technical aspects of localization are particularly scrutinized during certification testing for console games. Manufacturers of these gaming platforms impose strict and region-specific requirements on the technical aspects of localization. These include conformity with button naming conventions, error messaging formats, UI alignment, save/load prompts, and behavior under network disconnection. Failure to comply with these requirements may result in the game being banned from publication on a specific platform or delayed until all defects are resolved.

Before a game is released on consoles, it must undergo mandatory certification tests, which necessarily include:

- Evaluation of correctness in texts, the interface, or voiceovers, verifying that there are no display overflows, truncated strings, or incorrect speaker attributions.
- Verification of the proper adaptation of system notifications and the game’s user interface, including menu flow, confirmation dialogs, and region-appropriate terminology.

© International Software Testing Qualifications Board

- Verification of the functional correctness of the localized version according to the console's regional settings, including correct time/date formats, number systems, and language switching features across all supported regions.

7.3 Executing Localization Testing

In the process of localization testing, the following activities can be identified. Some of these may be conducted simultaneously:

- Analysis and planning of localization testing
- Execution of localization technical testing
- Linguistic testing and adaptation testing

Analysis and Planning of Localization Testing

This includes:

- Providing testers with all necessary documentation related to the test product.
- Analyzing the original version of the video game and any previously localized versions (if available).
- Creating a glossary and translation memory to help testers correctly interpret the terminology used.
- Compiling a database of game-related components and references to verify accurate translation and adaptation.
- Collecting materials describing all key characteristics of the target audience to assist in evaluating the adequacy of localization adaptation.
- Selecting and configuring defect management tools — a document or platform where all defects identified during localization testing will be recorded.

Execution of Technical Localization Testing

The following activities are performed:

- Verification of technical capabilities for internationalization.
- User interface testing.
- Subtitle testing (including synchronization with audio).
- Testing of regional language support (including text direction and special characters).
- Testing for missing texts and translations.
- Testing of audio and video content (including verifying that the audio corresponds to the relevant game components and the character's voice matches their gender).

Linguistic Testing and Adaptation Testing

These are the most complex and critical test types. They involve various tests carried out by a wide range of specialists. For example:

- Testing the functional correctness of the translation for all game content (including compliance with platform manufacturers' requirements, translation language rules, translation traditions).
- Testing the consistency of terminology.
- Verifying the correct usage of measurement units (including currency).
- Testing for stylistic and genre appropriateness (including references to source materials).
- Reviewing the game's visual, audio, and textual content for historical accuracy and alignment with real-world prototypes.

© International Software Testing Qualifications Board

- Testing the compliance of localized content with the norms and requirements of the target region.

In different companies, the phases of localization testing may vary, including or excluding specific tests or the even entire localization test effort depending on the situation. The test process described above serves as a basic framework and can be adapted to fit a specific project.

Additional tests may be conducted, such as:

- User interface testing
- Graphics testing
- Audio testing
- Certification testing

© International Software Testing Qualifications Board

8 Game Controllers Testing - 60 minutes

Keywords

Accessibility, compliance, ergonomics testing, interoperability, performance testing, usability

Video Game Specific Keywords

Gamepad, stick, video game controller

Learning Objectives for Chapter 8

8.1 Principles and Concepts of Game Controllers

GaMe-8.1.1 (K2) Distinguish input devices and their functions.

GaMe-8.1.2 (K2) Give examples of failures of video game controllers.

8.2 Test Types in Game Controller Testing

GaMe-8.2.1 (K1) Recall controller interoperability testing.

GaMe-8.2.2 (K1) Recall controller performance testing

GaMe-8.2.3 (K1) Recall controller ergonomics testing

GaMe-8.2.4 (K1) Recall controller compliance testing

GaMe-8.2.5 (K1) Recall controller usability testing

GaMe-8.2.6 (K1) Recall controller accessibility testing

8.1 Principles and Concepts of Video Game Controllers

8.1.1 Types of Video Game Controllers

A video game controller is an input device used in video games to control the movement and actions of game objects, typically a character (or one of the characters) in the game. Video game controllers provide more intuitive and precise control compared to standard keyboards and mice.

Below are the main types of game controllers:

Video Game Controller	Functionality
Gaming Keyboards and Mice	Specialized keyboards and mice with additional programmable buttons, high precision, and adjustable weight.
Gamepads (Joysticks)	The most common type of one-handed or two-handed controllers, featuring multiple buttons and analog sticks for movement and actions.

© International Software Testing Qualifications Board

Steering Wheels, Pedals, and Gear Shifters	Primarily used in racing games, providing a more realistic driving experience.
VR Controllers	Designed for use with virtual reality systems, allowing hand movement tracking and interactive capabilities in a virtual environment.
Specialized Controllers	Various devices such as dance pads, music controllers, and sports simulation devices designed for specific game genres.
Motion Controllers	Controllers equipped with accelerometers for motion tracking.
Touchscreens	An input and display interface that allows users to interact directly with on-screen content by touching the screen surface.
Adaptive Controllers	Devices designed for gamers with limited mobility.

The purpose of video game controllers is to provide smooth and intuitive control over the gameplay, enhance the user experience, and increase player enjoyment.

The main functions of a controller include:

- Providing interaction between the player and the game
- Enhancing convenience and ergonomics
- Increasing immersion and realism
- Supporting a variety of game genres
- Providing accessibility and inclusivity

Controller testing focuses on verifying how video game software interprets and responds to input from supported devices. From a game testing perspective, this includes validating input handling, responsiveness, configuration, and in-game behavior across supported platforms and control schemes.

Testing of controller hardware, firmware, or driver implementation itself is typically outside the scope of game testing. When input-related failures are observed, testers focus on determining whether the cause lies within the game software's handling of input or in external components, and report findings accordingly.

8.1.2 Failures Related to the Performance Efficiency of Video Game Controllers

It is important to note that this syllabus does not cover physical failures of game controllers caused by manufacturing defects or improper user handling.

Common failures include:

- Input lag
- Driver incompatibility
- Limited customization

© International Software Testing Qualifications Board

- Incorrect button mapping
- Failures with additional features (vibration, accelerometer functionality)

Input Lag

This refers to the delay between the moment a player performs an action (e.g., pressing a button or moving a stick) and when that action is displayed on the game screen. This delay is measured in milliseconds (ms) and directly affects control responsiveness. Such a failure is particularly noticeable in video games that require fast player reactions.

Driver Incompatibility

From a game testing perspective, driver incompatibility refers to situations where the video game software is unable to correctly interpret or respond to input provided through the controller's software interface. This may manifest as failure to recognize the device, incorrect operation of control elements, or input lag during gameplay.

While such failures may be influenced by external factors such as drivers, firmware, or operating system behavior, testing in this context focuses on identifying whether the observed failure originates from the game software's handling of controller input or lies outside the scope of the game application. Findings are reported accordingly, rather than attempting to validate or correct hardware, firmware, or driver implementations.

Limited Customization

This refers to situations where video game controller customization options are restricted due to software defects or driver failures preventing players from adjusting settings to their individual preferences.

Incorrect Button Mapping

Commands entered by the player are misinterpreted by the system. For example, pressing one button may trigger an action assigned to a different button, or the command may not register at all. This failure can be related to defective drivers or outdated controller firmware.

Failures with Additional Features (Vibration, Accelerometer Functionality)

The causes of such failures vary, including driver incompatibility, conflicts with other hardware or software, or firmware defects in the controller.

Typical failures of video game controllers are usually caused by outdated device drivers, conflicts with software from other connected devices, or firmware defects within the controller itself. These factors determine the approach to testing controllers.

8.2 Test Types in Game Controller Testing

Video game controller testing typically begins when the game functionality is ready for use with a standard game controller.

Game controller testing may include:

- Connecting/disconnecting the game controller to/from a PC or console.
- Using the game controller with a low battery level.
- Using one or multiple game controllers simultaneously.
- Unconventional use of the game controller (negative testing).
- Testing vibration functionality (presence and intensity level).

© International Software Testing Qualifications Board

- Verifying that supported versions/updates of the controller function correctly and that unsupported versions are detected and properly reported to the user.

8.2.1 Interoperability Testing

If an application requires the use of a game controller as an input device, testing must verify that all software functions, interface interactions, and gameplay mechanics must correctly interact with the connected device. The user must be able to control both the user interface components and the gameplay itself without switching controllers or experiencing lag in input recognition.

The assigned control components—buttons, analog sticks, steering wheel rotations, voice commands, or spatial movements—must correspond to the intended actions of the game character. This includes menu navigation, camera control, movement, combat actions, and inventory management. The tester must also verify that when the video game controller is disconnected from the device, the application pauses automatically if possible or displays a contextual prompt to inform the player.

If the game is implemented in real-time and cannot be paused, disconnecting the video game controller should not result in the player being disconnected from the server, dropped from a match, or losing progress. Instead, it should be accompanied by an informational message, giving the player an opportunity to reconnect the device or switch to an alternative control scheme without disrupting the session. Resilience, usability, and user experience continuity are critical criteria in this test phase.

8.2.2 Performance Testing

Performance testing of video game controllers focuses on measuring the responsiveness and reliability of input signals under various conditions. Key aspects include input latency—the time between a player's action (e.g., button press or joystick movement) and the corresponding on-screen response—as well as consistency in signal transmission over time. Performance testing is especially critical in fast-paced genres like fighting games, first-person shooters, or racing simulations, where even slight delays can affect gameplay, reaction timing, or competitive balance.

Testers assess the video game controller's behavior under different system loads, network conditions (for wireless video game controllers), and platform configurations to identify bottlenecks or irregularities in performance efficiency. Key performance indicators include frame delay, packet loss, signal interference, and input jitter. Additionally, testers must evaluate how quickly the system detects repeated inputs, such as rapid button presses (e.g., in quick-time events or combo sequences), and whether analog inputs like pressure sensitivity, dead zones, and tilt responsiveness remain stable over time.

Long-duration sessions are also tested to look for overheating, battery drain, input drift, or disconnect frequency, especially in Bluetooth or proprietary wireless video game controllers. Comprehensive performance testing evaluates if players experience fluid, predictable, and competitive gameplay, free of hardware-induced delays or inconsistencies.

8.2.3 Ergonomics Testing

The buttons on video game controllers from the most popular brands have established functions. The X, A, and □, X buttons on dual-handed video game controllers are very often used in applications as "confirm," "accept selection," or "interact" with an interactive component, while the B and ○ buttons function as "cancel," "decline," or "return." These conventions form a critical part of user interface consistency and player muscle memory, especially in genres like action-adventure, RPGs, and platformers.

Button combinations are configured and managed by software logic within the game engine. When testing button combinations, the tester must pay attention to whether they are anatomically comfortable

© International Software Testing Qualifications Board

for different groups of players (depending on genre, age, dominant hand, hand size, disability): whether the player can press multiple buttons simultaneously or sequentially without discomfort or misinputs.

Additionally, testers must assess the timing sensitivity of combos, the customizability of control schemes, and whether key remapping options are provided for accessibility. Games should also accommodate controller latency, verifying that multi-button actions register correctly across different hardware types, such as wired versus wireless controllers. Proper validation of button use and layout contributes to player comfort, accessibility, and inclusive design.

8.2.4 Testing Video Game Controllers for Compliance

The tester must validate that the developer uses traditional layouts for popular video game controllers, providing user familiarity and input consistency across genres. For example, as mentioned earlier, the X, A, and □, X buttons are used for "accept," while the B and ○ buttons function as "cancel." This convention is especially important for menu navigation, dialog confirmations, and interaction prompts.

On a keyboard, the W, A, S, and D keys are used for movement, Ctrl or C for crouching or crawling, and the spacebar for jumping. These mappings are standard in most PC games, especially first-person shooters, RPGs, and platformers.

The left mouse button is traditionally assigned for shooting, attacking, or selecting units in strategy games, while the right mouse button is typically reserved for aiming, camera control, or movement commands. The tester must verify any deviation from these norms is clearly documented, customizable, and intuitively introduced, especially for new players.

Incorrect mapping or lack of input flexibility can result in frustration, mistimed actions, and accessibility defects, particularly for players relying on muscle memory or assistive control schemes. Thorough testing verifies that control schemes meet user expectations and support a wide range of playstyles and skill levels.

8.2.5 Usability Testing

Usability testing for video game controllers focuses on evaluating how effectively and comfortably players can interact with the game using the controller. This includes assessing the intuitiveness of button layouts, ease of access to essential controls, and the overall ergonomics of gameplay interactions. Testers verify whether players can easily navigate menus, customize input settings, and perform in-game actions without confusion or discomfort. Special attention is given to accessibility features, such as remappable controls, support for alternative input devices, and usability by players with different physical abilities. The goal is to evaluate that the video controller provides a responsive, inclusive, and frustration-free experience across diverse player profiles and gaming scenarios.

Testers must verify that controller functions and button mappings are consistent with previous versions of the game. Any differences shall be clearly documented (ideally in-game) and implemented intuitively, with clear prompts and minimal penalty for unfamiliar inputs.

Testers must also verify that the use of any video game controller does not give players a significant advantage over others. Since dual-handed video game controllers are significantly inferior to the keyboard/mouse combination in terms of aiming speed and precision, they often include an aim assist feature that helps target enemies and track them automatically. The reticle (aiming mark or crosshair) simply "sticks" to the enemy. In this case, the tester's responsibility is to monitor how far the reticle follows the enemy and whether it locks onto their head, which is often a vulnerable area where a hit deals additional damage.

© International Software Testing Qualifications Board

In addition, testers shall assess how well the control scheme holds up in real gameplay contexts, including prolonged play sessions, high hand/finger workload, and stressful situations (e.g., combat, rapid action switching, pausing and one-handed interactions). Testers should validate that the control design does not cause frequent misinputs, unintended actions, or fatigue-related issues (e.g., the need to continuously hold a trigger/button, overly complex button combinations, or awkward confirmation flows in menus). The acceptance criterion is that controls remain predictable, easy to learn, and comfortable for a broad range of players, and that complex actions can be executed reliably without excessive strain or recurring frustration.

8.2.6 Accessibility Testing

Accessibility testing focuses on evaluating whether the game can be used by players with the widest possible range of characteristics and capabilities within a specified context of use (i.e., accessibility as a quality characteristic). Testers shall verify the presence and correct operation of accessibility capabilities that are commonly expected in modern games, including: color vision deficiency modes; configurable subtitles and/or captions (e.g., size, contrast, background, speaker identification); clear and scalable UI presentation; remappable controls; support for alternative input devices; and adjustable difficulty and assistance options that reduce cognitive and motor load. They should also validate support for text-to-speech narration for UI and dialogue where applicable;

Test planning should be informed by widely adopted industry accessibility guidelines used as best-practice benchmarks rather than strict pass/fail compliance rules and by the relevant platform submission and certification requirements, as these increasingly require demonstrable accessibility features and can introduce release risk if gaps are identified late. The acceptance criterion is that accessibility features are discoverable, configurable, and reliable across target platforms and the intended range of player capabilities, and that they materially reduce barriers without compromising core gameplay integrity.

© International Software Testing Qualifications Board

9 References

9.1 Standards

[ISO25000] ISO/IEC 25000:2014, Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE

[ISO25010] ISO/IEC 25010:2023, Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models

9.2 ISTQB documents

[ISTQB_AL_SEC] ISTQB Advanced Level Security Testing Syllabus, Version 2016

[ISTQB_ALTA_SYL] ISTQB Advanced Level Test Analyst Syllabus, Version 3.1.2

[ISTQB_ALTTA_SYL] ISTQB Advanced Level Technical Test Analyst Syllabus, Version 4.0

[ISTQB_ALTM_SYL] ISTQB Advanced Level Test Manager Syllabus, Version 2012

[ISTQB_CTFL_MAT] ISTQB Mobile Application Tester, Version 2019

[ISTQB_EXAM_S&R] ISTQB Exam Structure and Rules, Game Testing, Version 1.0

[ISTQB_FL_AT] ISTQB Foundation Level Agile Tester Syllabus, Version 2014

[ISTQB_FL_PT] ISTQB Foundation Level Performance Testing Syllabus, Version 2018

[ISTQB_FL_SYL] ISTQB Foundation Level Syllabus, Version 4.0

[ISTQB_GLOSSARY] ISTQB Glossary of Terms used in Software Testing, https://glossary.istqb.org/

[ISTQB_UT_SYL] ISTQB Foundation Level Usability Testing Syllabus, Version 2018

9.3 Books

[Nystrom14] Nystrom, R. (2014). Game programming patterns. Genever Benning., ISBN: 978-0990582908. URL: https://www.gameprogrammingpatterns.com/

[Buttfield19] Buttfield-Addison, P., Manning, J., & Nugent, T. (2019). Unity game development cookbook: essentials for every game. O'Reilly Media., ISBN: 9781491999158

[Lee16] Lee, J. (2016). Learning unreal engine game development. Packt Publishing Ltd., ISBN: 9781784398156

[Tavakkoli18] Tavakkoli, A. (2018). Game Development and Simulation with Unreal Technology. CRC Press., ISBN-13: 978-1498706247

[Romero19] Romero, M., & Sewell, B. (2019). Blueprints Visual Scripting for Unreal Engine: The faster way to build games using UE4 Blueprints. Packt Publishing Ltd., ISBN: 9781789347067

[Chandler11] Chandler, H. (2011). The Game Localization Handbook 2nd Edition, Jones & Bartlett Learning, ISBN: 0763795933

[Retsker81] Retsker, Ya. I. (1981). Textbook for translation from English into Russian. M.: Prosveschenie. (In Russian).

© International Software Testing Qualifications Board

[Torgovkin24] Torgovkin A. (2024). Video Game Testing. Bombora, ISBN: 978-5-04-196780-2

9.4 Links (Web/Internet)

Note: All references are current as of May 30, 2026.

- [ISTQB-Web] https://www.istqb.org/
- [URL1] https://research.ncl.ac.uk/game/mastersdegree/workshops/technicalrequirementschecklists/Technical%20Requirements%20Checklist%20Workshop.pdf
- [URL2] https://docs.microsoft.com/en-us/gaming/gdk/_content/gc/live/get-started/live-xbl-overview
- [URL3] https://igda-website.s3.us-east-2.amazonaws.com/wp-content/uploads/2021/04/09142137/Best-Practices-for-Game-Localization-v22.pdf

© International Software Testing Qualifications Board

10 Appendix A – Learning Objectives/Cognitive Level of Knowledge

The following learning objectives are defined as applying to this syllabus. Each topic in the syllabus will be examined according to the learning objective for it.

The learning objectives begin with an action verb corresponding to its cognitive level of knowledge as listed below.

Level 1: Remember (K1)

The candidate will remember, recognize and recall a term or concept.

Action verbs: Recall, recognize.

Examples
Recall the concepts of the test pyramid.
Recognize the typical objectives of testing.

Level 2: Understand (K2)

The candidate can select the reasons or explanations for statements related to the topic, and can summarize, compare, classify and give examples for the testing concept.

Action verbs: Classify, compare, differentiate, distinguish, explain, give examples, interpret, summarize

Examples	Notes
Classify test tools according to their purpose and the test activities they support.	
Compare the different test levels.	Can be used to look for similarities, differences or both.
Differentiate testing from debugging.	Looks for differences between concepts.
Distinguish between project and product risks.	Allows two (or more) concepts to be separately classified.
Explain the impact of context on the test process.	
Give examples of why testing is necessary.	
Infer the root cause of defects from a given profile of failures.	
Summarize the activities of the work product review process.	

Level 3: Apply (K3)

The candidate can carry out a procedure when confronted with a familiar task, or select the correct procedure and apply it to a given context.

Action verbs: Apply, implement, prepare, use

© International Software Testing Qualifications Board

Examples	Notes
Apply boundary value analysis to derive test cases from given requirements.	Should refer to a procedure / technique / process.
Implement metrics collection methods to support technical and management requirements.	
Prepare installability tests for mobile apps.	
Use traceability to monitor test progress for completeness and consistency with the test objectives, test strategy, and test plan.	Could be used in a LO that wants the candidate to be able to use a technique or procedure. Similar to 'apply'.

© International Software Testing Qualifications Board

11 Appendix B – Business Outcomes Traceability Matrix with Learning Objectives

This section lists the traceability between Certified Tester Game Testing Business Outcomes and Certified Tester Game Testing Learning Objectives.

Business Outcomes: Certified Tester Game Testing	GaMe-BO1	GaMe-BO2	GaMe-BO3	GaMe-BO4	GaMe-BO5	GaMe-BO6
GaMe-BO1 Describe basic concepts of video game testing	16					
GaMe-BO2 Determine video game risks, goals and requirements under the needs and expectations of stakeholders		12				
GaMe-BO3 Conceptually design, implement and execute basic video game tests			16			
GaMe-BO4 Know the approaches to video game testing and their purpose				9		
GaMe-BO5 Recognize the tools supporting video game testing					6	
GaMe-BO6 Determine how test activities align with the software development lifecycle and reduce the cost of developing and publishing video games						1

			GaMe-BO1	GaMe-BO2	GaMe-BO3	GaMe-BO4	GaMe-BO5	GaMe-BO6
1. The Specifics of Video Game Testing								
GaMe-1.1.1	K1	Recognize specifics of video game testing	x					
GaMe-1.1.2	K1	Recall product risks in video game software		x				
GaMe-1.1.3	K1	Recall how video game risks can be mitigated by testing.		x				
GaMe-1.1.4	K2	Give examples of specific failures related to video game testing		x				
GaMe-1.1.5	K2	Explain the specific test objectives of video game testing	x			x		
GaMe-1.2.1	K1	Recognize specific roles and tasks in the video game development team	x			x		
GaMe-1.3.1	K1	Recall test activities throughout the video game software development lifecycle						x
2. Testing Video Game Mechanics								
GaMe-2.1.1	K2	Differentiate the testing of gameplay mechanics and non-gameplay mechanics						
GaMe-2.1.2	K2	Differentiate the testing of core mechanics and meta mechanics	x					
GaMe-2.1.3	K2	Differentiate the testing of client, server, and client-server mechanics	x					
GaMe-2.1.4	K2	Give examples of failures in video game mechanics		x				
GaMe-2.2.1	K3	Apply test types and test approaches of video game mechanics testing			x	x	x	
3. Video Game Level Testing								
GaMe-3.1.1	K2	Distinguish the video game levels components	x					
GaMe-3.1.2	K2	Give examples of failures in video game levels.		x				
GaMe-3.2.1	K2	Summarize geometry testing.			x			
GaMe-3.2.2	K2	Summarize the playtesting.			x			

© International Software Testing Qualifications Board

GaMe-3.3.1	K3	Apply video game level testing.				x		
4. Graphics Testing								
GaMe-4.1.1	K1	Recall the types of video game graphic components	x					
GaMe-4.1.2	K2	Give examples of failures in video game graphics		x				
GaMe-4.2.1	K1	Recall artistic testing			x		x	
GaMe-4.2.2	K1	Recall technical testing of graphics.			x		x	
GaMe-4.2.3	K1	Recall playtesting.			x			
GaMe-4.3.1	K3	Apply graphics testing				x		
5. Audio Testing								
GaMe-5.1.1	K1	Recall the video game audio functions	x				x	
GaMe-5.1.2	K1	Recall the video game audio types	x					
GaMe-5.1.3	K1	Recall the video game audio techniques	x					
GaMe-5.1.4	K2	Give examples of failures in video game audio content		x				
GaMe-5.2.1	K2	Summarize content-audio testing.			x			
GaMe-5.2.2	K2	Summarize the main test types to technical testing of audio content.			x			
GaMe-5.3.1	K3	Apply audio testing				x		
6. AI and Physics Testing								
GaMe-6.1.1	K2	Distinguish the main aspects of video game AI	x					
GaMe-6.1.2	K2	Give examples of failures of video game AI		x				
GaMe-6.1.3	K2	Distinguish the main aspects of video game physics	x					
GaMe-6.1.4	K2	Give examples of failures of video game physics		x				
GaMe-6.2.1	K3	Apply video game AI testing				x	x	
GaMe-6.2.2	K3	Apply video game physics testing				x		
7. Localization Testing								
GaMe-7.1.1	K1	Recall the essentials of localization	x					
GaMe-7.1.2	K2	Distinguish the main types of video game localization	x					
GaMe-7.1.3	K2	Distinguish between full and partial localization	x					
GaMe-7.1.4	K1	Recall root causes of localization defects		x				
GaMe-7.1.5	K2	Give examples of failures of video game localization		x				
GaMe-7.2.1	K2	Summarize linguistic testing and adaptation testing			x			
GaMe-7.2.2	K2	Summarize localization technical testing			x			
GaMe-7.3.1	K3	Apply localization testing				x	x	
8. Game Controllers Testing								
GaMe-8.1.1	K2	Distinguish typical and specialized input devices and their functions	x					
GaMe-8.1.2	K2	Give examples of failures of video game controllers.		x				
GaMe-8.2.1	K1	Recall controller performance testing.			x			
GaMe-8.2.2	K1	Recall controller interoperability testing			x			
GaMe-8.2.3	K1	Recall controller ergonomics testing			x			

© International Software Testing Qualifications Board

GaMe-8.2.4	K1	Recall controller compliance testing			x			
GaMe-8.2.5	K1	Recall controller usability testing			x			
GaMe-8.2.6	K1	Recall controller accessibility testing			x			

© International Software Testing Qualifications Board

12 Appendix C – Release Notes

ISTQB® Certified Tester Game Testing Syllabus v2.0 is a major update. For this reason, there are no detailed release notes per chapter and section. However, a summary of principal changes is provided below.

This major release has made the following changes:

Syllabus structure

All learning objectives have been edited to make them atomic and to create one-to-one traceability from learning objectives to content to avoid having content without a corresponding learning objective. Each learning objective is described in a section with the same number (e.g., GaMe-1.1.1 is discussed in Section 1.1.1). The goal was to make this version more homogeneous, practice oriented and to be up to date in trends of industry (e.g., by including AI-related topics), easier to read, understand, learn, and translate, focusing on increasing practical usefulness and the balance between knowledge and skills. An important reason of the update is to align the content with the CTFL v4.0.1 which was published after CT-GaMe v1.0.1. All chapters are now organized in the following structure:

- Principles and concepts of game testing area,
- Test types for this testing area,
- Executing testing for this area.

There are 66 learning objectives in v2.0 compared to 55 in v1.0.1. The table below shows detailed statistics on the number of learning objectives and the training time.

Syllabus version	#K1 LO (time)	#K2 LO (time)	#K3 LO (time)	#Total LO (time)
Current (2.0)	20 (100 min)	25 (375 min)	7 (420 min)	52 (895 min)
Previous (1.0.1)	12 (100 min)	39 (585 min)	4 (240 min)	55 (925 min)

Changes in learning objectives

- The K2 GaMe-1.1.4 was transformed to “Give examples of specific failures related to video game testing” without changing K-level.
- Two K2 GaMe-2.1.1 and GaMe-2.1.2 were merged into one K2 GaMe-2.1.1 on gameplay mechanics and non-gameplay mechanics.
- The topic on approaches to testing video game mechanics (old K2 GaMe-2.2.2 and K2 GaMe-2.2.3) was split into two topics on test types on video game mechanics (K2 GaMe-2.2.1 and K2 GaMe-2.2.2) and video game mechanics testing (K3 GaMe-2.3.1).
- Chapter 5 on video game level testing was moved before chapter 3 on graphics testing and chapter 4 on audio testing.
- Old K2 GaMe-5.2.1 was replaced with K3 GaMe-3.3.1, old K2 GaMe-5.2.2 was removed.
- New sections 3.2.1 and 3.2.2 related to geometry testing and playtesting were added on game levels testing chapter.
- Old K2 GaMe-3.3.1 on executing graphics testing was replaced by K3 GaMe-4.3.1.
- Chapter on audio testing was renamed to “4. Audio testing”.
- Old K2 GaMe-4.3.1 on executing graphics testing was replaced by K3 GaMe-5.3.1.
- Chapter 6 on AI-related topics was added with appropriate new K1-K3 LOs.
- Old K2 GaMe-7.3.3 was split into three LOs: K2 GaMe-7.2.1 and GaMe-7.2.2, K3 GaMe-7.3.1.
- Old chapter 6 was renumbered to 8. Game Controllers Testing.
- Five K1 LOs were added: GaMe-8.2.1 to GaMe-8.2.6

© International Software Testing Qualifications Board

13 Appendix D – Index

3D model, 15, 32, 36, 42
accessibility, 6, 72
ambient, 45, 46
animation, 15, 16, 36, 38, 39, 41, 42, 47, 49, 54
artistic testing, 42
audio technical testing, 49
clipping, 15, 31, 44
collision, 19, 28, 31, 32, 35, 37, 54, 55
color coding, 28, 29
compatibility, 12, 43, 49
content-audio testing, 48, 49, 50
core mechanics, 21, 22, 24
cultural adaptation, 60, 63
defect, 14, 19, 21, 24, 29, 30, 48, 53, 54
distortion, 36, 46, 50
failure, 15, 19, 23, 28, 38, 39, 40, 48, 62, 69
functional correctness, 20, 21, 25, 65
game character, 28, 30, 32, 70
gameplay, 12, 13, 14, 15, 20, 21, 24, 27, 29, 30, 31, 32, 35, 37, 42, 43, 44, 49, 56, 69
gameplay mechanics, 18, 20, 21, 24, 27, 29, 45, 49, 55, 70
graphics technical testing, 36
hitbox, 37
internationalization, 11, 58, 61, 63, 65
level of detail (LoD)
  LoD, 39
Live Operations, 18
localization, 11, 13, 17, 36, 44, 49, 58, 59, 61, 62, 63, 64
mapping, 31, 36
meta mechanics, 21, 22, 25
narrative, 13, 15, 29, 49, 56
non-gameplay mechanics, 20, 21
occlusion, 37, 47, 48
pathfinding, 52
performance testing, 16, 32, 50, 70
playtesting, 14, 20, 25, 32, 42, 56
post-production phase, 18
pre-production phase,, 24
production phase, 24
rigid body physics
  RBP, 54
risk, 13, 14, 18, 72
root cause, 30
server-side mechanics, 22
setting, 26
soft body physics
  SBP, 54
sound effect, 47, 48, 49
sound zone, 50
stick, 69
structural geometry, 26, 27
terrain, 26, 27, 29, 30, 35
texture, 28, 36, 37, 39, 43
usability, 14, 15, 16, 20, 25, 70, 71
video game, 9, 10, 11, 12, 15, 16, 17, 35, 37, 56, 59, 60, 64, 77
video game controller, 67, 69, 70, 71
video game level, 11, 26, 27, 30, 31
video game mechanics, 11, 13, 16, 19, 23, 24, 26, 30, 32, 63
video game physics, 11, 57
visual effect, 40, 41
volume, 47, 48

© International Software Testing Qualifications Board