テスト技術者資格制度
Foundation Level シラバス
ゲームテスト

バージョン **1.0.1J01**

International Software Testing Qualifications Board

Provided by

Russian Software Testing Qualifications Board (RSTQB)

著作権について

改訂履歴

バージョン	日付	備考
v1.0.1	2022 年 10 月 21 日	GA リリース

JSTQB

バージョン	日付	備考
v1.0.1	2024 年 8 月 21 日	日本語版リリース

目次

謝辞

本文書は、RSTQB（Russian Software Testing Qualifications Board）のコアチームによって作成された：

Margarita Trofimova (Vice-president of the Board)
Alexander Alexandrov (Editor-in-Chief)
Vadim Lukovaty
Maksim Nikolaev
Pavel Sharikov
Alexander Torgovkin
Svetlana Yushina

コアチームは、レビューチームの提案と意見に感謝する。Russian Software Testing Qualifications Board は、本シラバスの作成に貢献した LLC "Bytex" に謝意を表する。

加えて、コアチームは、早期から継続的にご指導いただいたワーキンググループのリーダーおよびメンバーに謝意を表する：Galit Zucker（事務総長）、Graham Bath（専門家ワーキンググループ、議長）、Gary Mogyorodi（用語集ワーキンググループ、メンバー）、Klaus Skafte（試験ワーキンググループ、議長）

本シラバスまたはその前身のレビュー、コメント、投票に参加したのは以下の方々が参加した：

Ivan Pchelkin Natalia Sterkhova Irina Yakovleva
Matthias Hamburg Chunhui Li Blair Mo
Tal Pe'er Wim Decoutere Claude Zhang
Meile Posthuma Nishan Portoyan Francisca Cano Ortiz Wojciech Becla Jana
Gierloff Paul Weymouth
Nitzan Goldenberg Jürgen Beniermann Erwin Engelsma
Georg Haupt Gary Mogyorodi Péter Sótér
Bíró Ádám Darvay Tamás Béla Laura Albert
Gergely Ágnecz Graham Bath Mike Smith

この文書は、ISTQB® によって 2022 年 10 月 21 日に正式に発表された。

日本語訳については、Japan Software Testing Qualifications Board メンバーおよび以下の日本語翻訳ワーキンググループメンバーにより行われた。

日本語翻訳ワーキンググループメンバー：

小宮 鉄平（ポールトゥウィン）
佐野 あゆみ（グリー）
早川 彰彦
福田 圭佑（グリー）
丸井 瑞貴（ポールトゥウィン）
大段 智広（JSTQB）
堀米 賢　（JSTQB）
松木 晋祐（JSTQB）
吉澤 智美（JSTQB）

0. 本シラバスの紹介

0.1 本シラバスの目的

本シラバスは、ISTQB® 認定テスト担当者ゲームテスト資格のベースとなる。ISTQB® では、このシラバスを以下のように規定している：

1. 国内委員会（National Boards）は、その国の言語に翻訳し、トレーニングプロバイダーを認定する。国内委員会は、シラバスをその国の言語に適応させ、参考文献を修正することができる
2. 試験委員会に対しては、各シラバスの学習の目的に適応した試験問題を、その国の言語で作成すること
3. トレーニングプロバイダーに対しては、コースウェアを作成し、適切な教授法を決定する
4. 認定受験者に対し、（トレーニングコースの一環として、または単独で）試験準備を行う
5. 国際的なソフトウェアおよびシステムエンジニアリングのコミュニティに対し、ソフトウェアおよびシステムテストの専門性を向上させ、書籍や論文のテストベースとして提供する

0.2 認定テスト担当者 ゲームテスト

認定テスト担当者 ゲームテスト資格は、ソフトウェアテストに携わる人で、ゲームテストに関する知識を深めたい人、またはゲームテストの専門家としてのキャリアをスタートさせたい人を対象としている。また、ゲームソフトウェアエンジニアリングに携わる人で、ゲームテストへの理解を深めたい人を対象としている。

このシラバスでは、ゲームテストの主な側面について以下を考察する。

- 技術的な側面
- 方法論的な側面
- 組織的な側面

0.3 テスト担当者のキャリアパス

ゲームテスト資格は、Foundation Level を基礎として、テストのプロフェッショナルのキャリアパスの定義をサポートする。ゲームテスト認定資格の保持者は、Foundation Level で習得したテストに関する幅広い理解をさらに深め、テストのプロフェッショナルとしてゲームプロジェクトで効果的に働くことができるようになる。

ゲームテスト認定資格の保持者は、CT-GaMe の略称を使用することができる。

ISTQB® の最新のキャリアパスについては、[ISTQB-Web]を参照。

0.4 ビジネス成果

本節では、ゲームテスト資格認定取得者に期待されるビジネス成果を示す。

GaMe-1　ビデオゲームとゲームテストの基本概念を説明できる。
GaMe-2　利害関係者のニーズと期待の下で、リスク、ゴール、ゲームソフトウェアの要件を決定する。
GaMe-3　基本的なゲームソフトウェアのテストを概念的に設計、実装、実行できる
GaMe-4　ゲームソフトウェアのテストアプローチとその目的を知る。
GaMe-5　ゲームテストをサポートするツールを認識する。
GaMe-6　テスト活動がソフトウェア開発ライフサイクルとどのように整合するか、どのようにビデオゲームの開発とパブリッシングのコストを削減するかを判断する。

0.5 試験対象の学習の目的と知識レベル

学習の目的は、ビジネス成果の達成を支援するとともに、ゲームテストの認定試験の設計にも使用される。各学習の目的は、認知プロセス（K レベル）に言及している。

K レベル（認知レベル）は、Bloom [Anderson01]の改訂版分類法に従って学習の目的を分類するために使用される。ISTQB® では、この分類法を用いてシラバスに基づいた試験を設計している（詳細は付録 A を参照）。

本シラバスは、3 つの異なる K レベル（K1 から K3）を考慮している。

K レベル	キーワード	説明
1	記憶	受験者は用語や概念を覚えているか、認識していなければならない。
2	理解	受験者は、設問のトピックに関連する記述の説明を選択できなければならない。
3	適用	受験者は、ある概念や手法の正しい適用を選択し、それを与えられた状況に適用しなければならない。

基本的に、この教育プログラムのすべてのパートには、レベル K1 の学習の目的が含まれている。つまり、認定候補者は用語や概念を認識し、記憶し、想起しなければならない。レベル K2 および K3 の学習の目的は、それらを含む章の冒頭に示されている。

0.6 認定テスト担当者 ゲームテスト資格試験

認定テスト担当者ゲームテスト資格試験は、本シラバスに基づいて実施される。試験問題の解答には、本シラバスの複数のセクションに基づく資料の使用が要件となる場合がある。シラバスの「本シラバスの紹介」と「付録」を除くすべての章が試験対象となる。標準や書籍は参考文献として含まれるが、それらの内容は、シラバス自体にそのような標準や書籍から要約されているもの以外、試験対象にはならない。

認定テスト担当者ゲームテスト資格試験の構成とルールについては、[ISTQB_EXAM_S&R] を参照。

ISTQB® Foundation Level 認定 [ISTQB_FL_SYL] は、認定テスターゲームテスト認定試験を受験する前に取得しなければならない。ただし、受験者には次のことも強く推奨する：

- ゲームソフトのユーザー受け入れテスト担当者として 1 年以上の経験があるなど、ゲームソフト開発またはテストに関する最低限のバックグラウンドがあること
- ISTQB® のコース（国内委員会により認定されたもの）を受講する

0.7 認定審査

ISTQB®のメンバー委員会にて、トレーニングコースの教材が本シラバスに従っているトレーニングプロバイダーを認定する。トレーニングプロバイダーはメンバー委員会または認定を行う機関から認定ガイドラインを入手しなければならない。トレーニングコースがシラバスに従っていると認定されると、トレーニングコースの一部として ISTQB®の試験を実施することができる。本シラバスの認定ガイドラインは、プロセスマネジメント・コンプライアンスワーキンググループが発行する一般的な認定ガイドラインに準ずる。

0.8 標準の取り扱い

認定テスト担当者ゲームテストシラバスにおいて参照される標準がある（例えば IEEE、ISO など）。これらの参考文献は、（品質特性に関する ISO 25010 の参考文献のように）枠組みを提供し、読者が望む場合には追加情報のソースを提供するためのものである。標準ドキュメントは試験を目的としたものではない。標準の詳細については「第 8 章.参考文献」を参照。

0.9 最新情報の維持

ソフトウェア業界は急速に変化している。これらの変化に対応し、関係者が関連する最新の情報にアクセスできるように、ISTQB ワーキンググループは、www.istqb.org ウェブサイトにリンクを設け、サポートドキュメントや標準の変更点などを紹介している。これらの情報は、本シラバスでは試験対象外である。

0.10 詳細レベル

本シラバスの詳細レベルは国際的に一貫したトレーニングと試験を可能にする。このゴールを達成するために本シラバスは以下の内容で構成されている。

- 認定テスト担当者ゲームテストの意図について説明する全般的な指導の目的
- 想起することができなければならない用語（キーワード）のリスト
- 達成すべき認知的な学習の成果について説明している、知識領域ごとの学習の目的
- 認められた情報源の参照を含む主要なコンセプトの説明

シラバスの内容は、ゲームテストの知識分野全体を記述したものではなく、認定テスト担当者トレーニングコースでカバーされる詳細レベルを反映したものである。ほとんどのゲームプロジェクトに適用できるテストの分野と技法に焦点を当てている。

0.11 シラバスの構成

認定テスト担当者ゲームテストのシラバスには、ゲームテストのスペシャリストになるために必要な知識を網羅した 7 つの章がある。

各章のトップレベルの見出しには、その章の最低時間が明記されている。認定トレーニングコースの場合、シラバスでは最低 15 時間 25 分の授業が要件とされ、以下のように 7 つの章に配分されている。

- 第 1 章: 75 分　ゲームテストの特異性

- ゲームソフトウェアのテストに関する基本原則、ゲームテストが必要とされる要件、テスト目的、ゲームテストとゲームプレイの違いなどを学習する
- テスト担当者は、ゲームテストのプロセス、主な活動、作業成果物を理解する

- 第 2 章: 180 分　ゲームメカニクスのテスト
  - テスト担当者は、異なるゲームメカニクスとそのテストアプローチを区別することを学ぶ
  - テスト担当者は、ゲームメカニクスのテストに使用される動的および静的メソッドの両方を学ぶ
- 第 3 章: 165 分　グラフィックステスト
  - テスト担当者は、ゲームプロダクトのグラフィックコンテンツの特徴やタイプ、ツールサポートを含むグラフィックステスト実行の良い実践例について学ぶ
  - テスト担当者は、関係者の責任範囲について学ぶ
- 第 4 章　190 分　サウンドテスト
  - テスト担当者は、ゲーム製品のサウンドコンテンツのタイプや技術、テスト実行のベストプラクティス（ツールサポートを含む）について学ぶ
  - テスト担当者は、関係者の責任範囲について学ぶ
- 第 5 章: 65 分　ゲームレベルテスト
  - テスト担当者は、レベル設計とテストアプローチがゲームのジャンルとそのツールサポートに依存することを学ぶ
  - テスト担当者は、関係者の責任範囲について学ぶ
- 第 6 章: 95 分　ゲームコントローラーのテスト
  - テスト担当者は、関連する欠陥やその可能性のある原因を含め、ゲームコントローラーの種類について学ぶ
  - テスト担当者は、コントローラーのテストアプローチとそのツールサポートについて学ぶ
- 第 7 章　155 分　ローカライゼーションテスト
  - テスト担当者は、ローカライゼーションとインターナショナリゼーションの基本や違い、そのリスク、テストへのアプローチ、ツールのサポートについて学ぶ
  - テスト担当者は、関係者の責任範囲について学ぶ

1. ゲームテストの特異性 - 75 分 (K2)

テストキーワード
機能テスト、プレイテスト

ビデオゲームに特化したキーワード
3D モデル、コンセプト段階、ビデオゲームデザイナー、マルチプラットフォーム、ポストプロダクション段階、プリプロダクション段階、プロダクション段階、ビデオゲーム

第 1 章の学習目標

1.1 ゲームテストの基本
GaMe-1.1.1 (K1) ゲームテストの目的と特徴を理解する。
GaMe-1.1.2 (K2) ゲームソフトのプロダクトリスクについて例を挙げる。
GaMe-1.1.3 (K2) ゲームテストに関する欠陥について例を挙げる。
GaMe-1.1.4 (K2) ゲームテストのリスクを軽減する方法を要約する。
GaMe-1.1.5 (K2) ゲームのテストとプレイを比較する。

1.2 ゲーム開発チームの典型的な役割
GaMe-1.2.1 (K1) ゲーム開発チームにおける具体的な役割とテスト作業を識別する。

1.3 ゲームソフトウェア開発ライフサイクルを通じたテスト活動
GaMe-1.3.1 (K1) ゲームソフトウェア開発ライフサイクルを通してテスト活動を想起する。

1.1 ゲームテストの基本

1.1.1 ゲームテストの特徴

ソフトウェアテストでは、さまざまな専門分野を区別するのが通例である。アプリケーションの性能テストに従事するテスト担当者もいれば、モバイルアプリケーションのテストに注力するテスト担当者、セキュリティシステムの脆弱性を探すテスト担当者、コンピューターゲームの欠陥を探すテスト担当者もいる。

ゲームのジャンルはさまざまであり、1 人または複数のプレイヤーを対象としている場合もある。しかし、ゲームのテストのアプローチは、これらの特徴から独立していると考えることができる。本シラバスのセクション 2.1 では、ゲームメカニクスの機能テストアプローチと非機能テストアプローチについて詳述する。

本シラバスでは、"ゲーム"、"コンピューターゲーム"、"ビデオゲーム" という用語は同義語として扱われることに注意することが重要である。これは、文章を読みやすくし、シラバスの文脈に合わせるためである。ゲームとは、あくまでもソフトウェアであり、スポーツの試合やギャンブルと混同してはならない。

1.1.2 ゲームプロダクトのリスク

ビデオゲーム産業の爆発的な発展により、さまざまなプラットフォームで配信されるようになった。ゲームは大型化し、複雑度を増し、技術的リソースへの要求も厳しくなっている。同時に、プレイヤーの層も大幅に増加し、より洗練され、ゲームの品質に対する要求も厳しくなっている。

一般的なソフトウェアプロジェクトやプロダクトのリスクは、ビデオゲームやゲーム開発にも当てはまる。その結果、ゲームも他のソフトウェアと同様にテストの対象となる。いくつかのリスクはゲーム分野のソフトウェアに特有であり、より注意を払う必要がある。そのようなリスクには以下のようなものがあるが、これらに限定されるものではない：

- 不正行為による不当な利益の獲得
- プレイヤーの主観による市場での成功に対する依存
- マルチプラットフォームの課題
- ゲームメカニクスの有効性を予測する複雑度（2.1 節参照）

テスト担当者は、次のような箇所で欠陥を発見することができる：

- 開発ライフサイクルの初期段階におけるゲームソフトウェアのアーキテクチャ
- サウンドデザイン
- すでに発売されているゲームのテキスト

1.1.3 ゲームテストに関する欠陥

ビデオゲームで特によく見られる欠陥がいくつかある。以下はその例である：

- グラフィックとアニメーションの問題
- ビデオゲーム世界の物理法則の違反
- ビデオゲームの人工知能の欠陥
- ビデオゲームのプロット構築における不正確さ
- 一部のインターフェース要素の誤った機能(ビデオゲームのキャラクターが空中から出現する、敵を攻撃しない、ホバリングする車など)

このような欠陥は発見しやすいかもしれないが、それを再現するための正確な手順を特定することは、しばしば困難である。この複雑度が、ゲームテストを難しいものにしている。優れたテスト担当者は、意図されたゲームシナリオに沿ってテストを設計し、それが仕様書を満たしていることを検証し、設計されたシナリオからの逸脱の可能性を分析し、逸脱の結果についてレポートしなければならない。

ネガティブテスト

ゲームをテストする際には、他のソフトウェアと同様に、明示的な欠陥を特定することが重要であり、さらにはユーザーが行った標準的でない行動の結果として現れる暗黙の欠陥を発見するよう努める。例えば、敵と戦ったり、鍵のかかったドアの鍵を探したりする代わりに、ユーザーはマップ全体から箱を集め、それらではしごを作り、柵を乗り越えたりすることで、ゲームで意図していた厳しい戦いを避けることができる。

また、ゲームに使用される建物の 3D モデルなど、他のオブジェクトとの相互作用が正しく設定されていない場合にも欠陥が発生する可能性がある。この種の欠陥は、プレイヤーが操作するキャラクターが壁を通過することを誤って許可する可能性があり、ユーザエクスペリエンス全体を損ない、プレイヤーに不正なゲーム上の利点を与える可能性がある。これは特にマルチプレイヤーゲームでは致命的である。このような壁の後ろに隠れると、プレイヤーは対戦相手から隠れることができるが、対戦相手を撃つことはできる。

このようなリスクは、ゲームプロジェクトのネガティブテストの重要性を高める。開発者が意図したとおりにプレイしようとせず、より簡単に、より少ない労力で勝つために、システムを「破る」または迂回しようとするプレイヤーが多数存在する可能性がある。システムを破ったり迂回したりするためには、必ずしもサードパーティのソフトウェアや特別なコマンドを使う必要はない。「明示的な」欠陥の発見だけで十分な場合もある。そのようなプレイヤーは、ゲームのロジックの欠陥を意図的に探し、無限の資源や強力な武器を手に入れるなど、個人的な利益のためにそれを利用する。収益化されたゲームでは、これはゲーム運営者の利益を減らすことにもつながる。

プレイヤーの意見への依存

ゲームソフトは、他のソフトウェアと同様に、エンドプレイヤーの主観的で感情的な判断に左右されることが多い。ビジネス上の問題を解決するために設計されたソフトウェアの場合、その機能性が最も重

要な要素となる。ゲームソフトの場合、最も重要な要素はユーザーの関心度とゲームセッションの印象である。機能的欠陥については目をつぶることができるかもしれないが、ゲームが退屈で単調であったり、時代遅れのグラフィックを使用していたりすると、ユーザーは簡単に使用をやめてしまうかもしれない。ビデオゲームの利用者の大部分は、主にゲーム評論家、レビューア、その他のオピニオンリーダーからのフィードバックに基づいて、製品の購入や使用を判定する。

マルチプラットフォーム

ゲームソフトウェアの重要な特徴は、複数のプラットフォームで使用されることが多いことである。できるだけ多くのユーザーに楽しんでもらうため、ゲームスタジオやパブリッシャーは、さまざまなパーソナルコンピューター（PC）ビルド、ウェブ、モバイルデバイス、ゲーム機など、多種多様なプラットフォームでリリースしている。

すべてのアップデートは、利用可能なプラットフォームごとにテストしなければならない。これは深刻なリスクをもたらし、テストに要する時間を大幅に増加させる。

中型のパソコンではよく動くゲームでも、最新世代のゲーム機ではさまざまな欠陥が発生する。また、さまざまな技術間の非効率的な通信や、もともとゲームが開発されたプラットフォームから、開発委託先が別のプラットフォームにゲームを移植するための要件が不完全なために欠陥が発生する可能性もある。

このようなリスクを軽減するためには、ゲームソフトウェアの開発とテストに多くの時間を割き、多くの異なるハードウェア構成でゲームの機能と性能をテストし、各プラットフォーム固有の特徴に基づいてテストを実施することが推奨される。

ゲーム機でのビデオゲームのテストは、特に考慮すべき点である。ゲーム機はゲーム専用に設計された機器であり、モバイル機器やコンピューターのように他のソフトウェアは含まれていない。

ブラックボックステスト技法（black-box test technique）を使ったテストでは、プラットフォームによる違いはほとんどない。しかし、各ゲーム機メーカーは、ビデオゲームが公開される前に準拠しなければならない独自の要件を定めている場合がある。これらの要件は、秘密保持契約に基づいて開発者やパブリッシャーに提供される独自の文書である。各要件のチェックリストはいくつかのカテゴリーで構成されており、ゲーム機メーカーに拒否されないためには、ゲームソフトウェアはそれらに準拠しなければならない[URL1], [URL2]。

従って、ゲーム機でテストするのテスト担当者は、標準的なソフトウェアテスト手法に加えて、これらの要件に準拠しているかどうかをテストする必要がある。

欠陥を検出し、特定し、その理由を理解するために、特別な機器を使用する必要がある場合がある。この装置は基本的に同じゲーム機だが、ゲームの開発とテストに役立つ追加モードを提供する。このゲーム機は、承認された開発者およびテスト担当者用のサイトに登録され、その後認証が行われ、特別なモードへのアクセス権限が付与される。

1.1.4　ビデオゲームのプロダクトリスクを軽減するテストとは？

ほとんどのビデオゲーム、特にオープンワールドゲームは複雑度が高く、ゲームオブジェクト、イベント、要素の可能な限りの組み合わせをテストすることは不可能である。ソフトウェア開発ライフサイクル全体を通して、各組み合わせの単体テストを実施するのでさえ、長い時間がかかる可能性があり、ゲームの次のバージョンでの変更に対する確認テストは、追加の労力を必要とする。

リスクマネジメントは、リソースを効率的に配分し、起こりうる欠陥をその発生の可能性とゲームへの影響に基づいて評価するために用いられる。

各テストケースに対して、テストの優先度を決定するために 2 つの質問がなされる：

1. プレイヤーがここで欠陥を発見する可能性は？
2. もし欠陥が発見された場合、プレイヤーやゲームを所有する会社にどのような影響を与えるか？

例えば、あるゲームが直線的な構成であり、ユーザーがそのコンテンツに段階的にアクセスする場合、最初のゲームセッションで発生した欠陥は、おそらく 50 時間目のプレイで発生した欠陥よりも修正すべき優先度が高くなる（ただし、必ずしも重要度が高いとは限らない）。ロイヤルユーザーは、すでにゲームに多くの時間を費やしている場合、遭遇した欠陥をより冷静に受け止める可能性が高く、ゲームに対するユーザーの全体的な印象に大きな影響を与えない。このような欠陥が、導入トレーニング中にすべての新規プレイヤーに影響を与える場合は、また別の問題である。この場合、多くのプレイヤーはそのゲームプロダクトを拒否するかもしれない。

テストによってリスクを減らすには、ゲーム内の個々の欠陥を探すのではなく、欠陥のグループを特定することが重要である。これにより、開発者はゲームのコードやアーキテクチャの問題を同時に解決し、最終製品の品質向上を提供することができる。

欠陥グループとは、重要な領域における欠陥のクラスターである。ゲームプロダクトには、グラフィック、サウンド、ローカライゼーション、クライアント・サーバー・インタラクション、ハードウェアなど、欠陥が発生しやすい領域がある[Nystrom14]。

プレイヤーのロイヤリティを高めるには、ゲームのプレイを学んだり、チュートリアルを受けたりする段階が重要である。この段階では、プレイヤーが製品を評価できるように、膨大な数のゲーム機能が利用可能となる。そのため、この初期段階は非常に重要であるとされている。

1.1.5　テストと "プレイ" の違い

コンピューターゲームのテスト担当者の仕事の本質は、同僚と一日中オンラインで遊ぶことだという印象を持つかもしれない。しかし、この見解は真実からかけ離れている。

テスト担当者は、新しいプロジェクトを知るときやプレイテストのときを除いて、ただプレイするだけということはほとんどない。テスト担当者の仕事のほとんどは、さまざまなタイプのテストを実施することであり、それは多くの点で、他のソフトウェアに使用されるテストのタイプと共通している。

一般的なユーザーは、クリアするため、対戦相手に勝つため、あるいは楽しい時間を過ごすためにゲームを始めるのに対し、テスト担当者は、ゲームが仕様書に書かれた要件を満たしているかどうかを検証する。さらに、テスト担当者の個人的な嗜好は、ゲームそのものが意図するターゲット層と一致しないかもしれない。それにもかかわらず、テスト担当者は、例えばゲーム対象が正しく表示されているかどうかをテストするために、同じステージを何度も何度も繰り返すことになる。テスト担当者がこれらの反復作業を行う際に感じる単調さは、開発者が考案したゲームプレイを妨げる可能性のある欠陥を発見することで得られる満足感によって補われる。

他のシラバスとの関係

このシラバスで説明するアプローチは、ゲームテストにのみ適用される。ソフトウェアテスト（例：ゲームメカニクスのテスト）とハードウェアテスト（例：コントローラーのテスト）の両方に適用される。

特に、リグレッションテスト、性能テスト、使用性テストのようなテストタイプを実行する場合、ゲームのテストには、他のソフトウェアのテストとそれほど変わらない側面もある。基本的なテストコンセプト、非機能テスト技法、およびモバイルプラットフォームでのソフトウェアテストの特徴については、他の ISTQB シラバスで詳しく取り上げている。

1.2　ゲーム開発チームの典型的な役割

ほとんどの場合、コンピューターゲーム開発チームには、ソフトウェア開発の他の分野で見られる役割と基本的に変わらない役割が含まれる（ただし、これに限定されない）。ビジネスオーナー、プロジェ

クトマネージャー、アナリスト、開発者、ゲームデザイナー、テスト担当者などである。小規模なチームでは、1 人が同時に複数の役割を担うこともある。以下は、ゲームソフトウェアプロジェクトにおける典型的な役割の主な機能をまとめたものとなる。ゲーム開発の特殊な分野でチームに含まれる可能性のあるその他の役割については、本シラバスの該当する章に記載している。

役割	責任
経営者	● ビジネス開発 ● マーケティング方針の決定 ● 新製品の開発 ● チームビルディングに参加する
プロジェクトマネージャー	● タスクの設定と実行の監視 ● ソフトウェア開発ライフサイクルの各段階を把握する
アナリスト	● 製品要件を特定する ● オリジナルの仕様書に基づいて作業する ● 仕様書を更新し、常に最新の状態に保つ
開発者	● コードの開発とデバッグ ● 欠陥の修正 ● コンポーネントテストを開発
ゲームデザイナー	● ゲームメカニクス、バランス、プロット、レベル、ゲーム経済をデザインする ● ゲームドキュメントの作成 ● ゲーム内容を熟考し、発展させる
テスト担当者	● テストモデルとテスト戦略の開発 ● テストデータの設計と準備 ● 仕様書のレビュー ● 各種テストの実施 ● 欠陥の発見と分析

1.3 ゲームソフトウェア開発ライフサイクルを通じたテスト活動

コンセプト段階

この段階の主な目的は、開発チームと顧客の間で、将来の製品に関する共通のビジョンを達成することである。アイデアは、ゲームの一般的なビジョン、ジャンル、設定、ゲームプレイの本質、メカニズム、主な機能を簡潔に記述した文書の形で作成される。

この段階でレビューされる可能性のある成果物には、以下のようなものがある：

- ビジョンドキュメント
- コンセプトドキュメント
- 製品および技術的制約

この段階で作成されるテストウェアは以下のとおり：

- テスト戦略
- テスト計画の草案
- 高レベルのテスト工数見積り

- テスト環境

プリプロダクション段階

この段階の主な目標は、製品の初期作業バージョンである将来のゲームのプロトタイプを作成することである。プロトタイプは、初期品質で主要なゲームプレイを実装する。この段階では、中核となるゲームメカニクスと初期の仮説がテストされ（第 2 章参照）、チームの技術的能力が評価される。テスト担当者、ゲームデザイナー、開発チームの他のメンバーなどがプロトタイプのテストに参加できる。

開発チームは、実装の可能性を確立するために要件案をレビューし、テストチームは、メカニクス同士の相互作用や組み立てられたプロトタイプの機能を特定するために要件をレビューする。さらに、テスト担当者はテスト計画書とテストスケジュールのドキュメント作成にも関与する。

ゲームのドキュメントが作成されると、テスト計画書が作成・更新され、ドキュメントのレビューが行われる。このような尺度が、プリプロダクション段階におけるプロダクトの技術的リスクを軽減する。

この段階でレビューされる可能性のある成果物には、以下のようなものがある：

- 要件仕様書
- ゲームのプロトタイプ
- 前ステージで作成されたテスト成果物

この段階で作成・更新されたテストウェアは以下のとおり：

- テスト戦略
- テスト計画書
- テストスケジュール
- テスト環境

プロダクション段階

プロダクション段階ではゲームプロダクトが作成される。これは最も長いステージであり、通常はアルファ版とベータ版に分かれる。

この段階で、テスト担当者はチェックリストとテスト条件一式を確定し、実装されたプロダクトコンポーネントの機能テストと非機能テストを実行する。テストレベルとしては、コンポーネント統合テスト、システムテスト、受け入れテストが実施され（[ISTQB_FL_SYL]を参照）、最も多くの欠陥が特定されるため、テスト担当者は確認テストとリグレッションテストを実施することになる。

この段階で作成または更新されるテストウェアは以下のとおり：

- テスト計画
- テストスケジュール
- テストケース、テスト条件、チェックリスト、テストスイート、スクリプト
- 欠陥レポート
- テストレポート

ポストプロダクション段階

ポストプロダクション段階のゲームのテストには、ソフトウェアのメンテナンス期間中の一般的なテスト活動が含まれる。これには通常、[ISTQB_FL_SYL]で説明されているように、アップデートのテストとリグレッションテストが含まれる。

2. ゲームメカニクスのテスト - 180 分 (K3)

テストキーワード
アドホックテスト、機能テスト

ビデオゲームに特化したキーワード
クライアントメカニクス、コンセプト段階、コアメカニクス、ビデオゲームデザイナー、ビデオゲームメカニクス、メタメカニクス、プリプロダクション段階、プロダクション段階、サーバーメカニクス、ビデオゲームステート

第 2 章の学習目標

2.1 ゲームメカニクス
GaME-2.1.1 (K2) ゲームの仕組みの種類を分類する。
GaME-2.1.2 (K2) ゲームプレイメカニクスのテストと非ゲームプレイメカニクスのテストを区別する。
GaME-2.1.3 (K2) コアメカニクスとメタメカニクスのテストを区別する。
GaME-2.1.4 (K2) クライアント、サーバー、クライアントサーバーのメカニクスのテストを区別する。
GaME-2.1.5 (K2) ゲームメカニクスの欠陥の例を挙げる。

2.2 ゲームメカニクスのテストアプローチ
GaME-2.2.1 (K2) ゲーム制作のさまざまな段階における主なアプローチとテスト対象を要約する。
GaME-2.2.2 (K2) ゲームメカニクスをテストすることの重要性を理解する。
GaME-2.2.3 (K2) ゲームメカニクスを説明した文書をレビューすることの重要性を理解する。
GaME-2.2.4 (K3) ゲームメカニクスのテストの基本的なアプローチを適用する。

2.1 ゲームメカニクス

2.1.1 ゲームメカニクスの種類

第 1 章では、ゲームにとって最も重要なことの 1 つは、ユーザーを維持し増やすために、ユーザーの興味を喚起することであると述べた。プレイヤーはゲームのグラフィック、ゲームプレイそのもの、そしてゲームプレイを構成するメカニクスを評価する。

ビデオゲームのメカニクスは、あらゆるゲームの中心である。ビデオゲームのメカニクスとは、ゲームとユーザーの相互作用の仕組み的なものである。指定された要件内で、ゲームとユーザー間の影響やフィードバック、ビデオゲームの状態の変化などを考慮する。メカニクスは、ゲームのさまざまな側面や目標によって、さまざまなタイプに分けることができる。

プレイヤーとメカニクスの相互作用のタイプによる分類：

- ゲームプレイメカニクスは、ユーザーが意識的にゲームシステムと対話し、情報の可用性に基づいて行動する場合に関連する。ユーザーは、ビデオゲームの状態の変化に影響を与えることができる
- 非ゲームプレイメカニクスとは、ユーザーが自分の行動によってゲーム環境の状態に影響を与えることができないか、部分的にしか影響を与えることができない場合に関連する。ゲームメカニクスの一部が隠されているため、ユーザーは自分の影響を認識できない

主なゲームプレイへの影響による：

- コアメカニクスとは、ゲームの根幹をなすもので、ユーザーがゲーム中に繰り返すアクションを定義する。これらは、クリエイターによってゲームに組み込まれた特定のエクスペリエンスをユーザーが得られるようにすることを目的としている

- メタメカニクスはメインのゲームプレイの外にあるが、ゲームプレイに影響を与えることができる。それらは、ゲームデザイナーが望むこと（例えば、ゲームに戻る、プレイを続ける、購入をする）をプレイヤーにさせることを目的としている。これらのメカニクスは、コアメカニクスと組み合わせることで、ゲームをより面白くすることができる

ゲームの構造のアーキテクチャによる：

- クライアントメカニクスのユーザーのアクションの処理は、ユーザーのデバイスのみで行われる
- サーバーメカニクスはゲームサーバーでのみ処理されるメカニクスに関連する
- クライアントサーバーメカニクスは、ユーザーのデバイスとサーバーの両方で処理されるメカニクスに関連する。この場合、サーバーとクライアントの間で常にデータのやりとりが行われる

2.1.2 ゲームプレイメカニクスのテストと非ゲームプレイメカニクスのテストの違い

ユーザーは、ゲームに実装されているすべての仕組みに直接触れるわけではない。

ゲームプレイのメカニクスは、公式または非公式の仕様書に記述されるかもしれないが、ゲームプレイ以外のメカニクスは、ユーザーに対して記述されなければならない。多くの場合、それらはゲームプレイの目標達成に直接影響する。例えば、ゲームキャラクターによるコイン拾いは、限られた時間内に行わなければならない。コインを拾い、コインの数を増やす動作の機能は、ゲームメカニクスの一例である。

非ゲームプレイメカニクスは、ユーザー自身の行動によって引き起こされることもあるが、通常はユーザーから隠されている。例えば、ユーザーがゲーム内で商品を購入すると、非ゲームメカニクスのシーケンシャルフローが開始される。一例を以下に記載する。

1. ユーザーのゲームアカウントの状態を取得するために、データベースへのリクエストが行われる
2. 資金を引き出すための取引が行われる
3. 購入はログに記録され、実行されたアクションに関する情報は特別なテキストファイルに記録される
4. 購入の詳細が記載されたメールがユーザーに送信される

さらに、購入日や購入した商品の種類を記録しておくことで、類似商品の割引に関するメールを自動送信することができる。非ゲームプレイのメカニクスは明らかではないため、正式な要件として指定する必要がある。

ゲームプレイと非ゲームプレイのメカニクスのテストは、機能テストである、[ISTQB_FL_SYL] に記述されている。

2.1.3 コアメカニクスとメタメカニクスのテストの違い

コアメカニクスとメタメカニクスのテストアプローチの違いは、その影響力によって異なる。

ゲームから核となるメカニクスを取り除くと、ゲームプレイは完全に変わってしまう。例えば、レーシングシミュレーターから車の動きがなくなれば、そのゲームはもはや「レーシング」とは分類できない。

メタメカニクスの例としては、レースコースのタイプに応じて特定のパラメータを持つ車種を選んだり、戦闘に入る前にプレイヤーキャラクターに新しい防具を購入して装備させたりすることが挙げられる。ゲームからメタメカニクスを削除しても、ゲーム全体の本質が変わることはほとんどない。

機能要件に加えて、コアメカニクスとメタメカニクスには、それらがどのようにテストされるべきかに直接影響する非機能要件がある場合がある。

例えば、プレイアブルキャラクターの外見を変えるメタメカニクスにとって、最も重要なことは、それが正確に機能することである。たとえこのメカニクスが、プレイヤーがボタンを押してから数秒後に外見が変わるような、わずかな遅延を伴って機能したとしても、テスト担当者もユーザーもこれを欠陥と

は考えないだろう。しかし、キャラクターがジャンプしたりキャラクターがジャンプしたり、バトルでアビリティを使ったりするようなコアなメカニクスでは、このような遅延は致命的であり、プレイヤーの勝利を妨げかねないからである。その結果、性能テストはコアメカニクスにとっても重要である。

コアメカニクスとメタメカニクスの本質は、それらがテストされるタイミングにも影響する。原則として、それらの検証は通常、ゲーム開発の異なる段階で開始される。ゲームキャラクターの動きや報酬の受け取りなど、コアとなるメカニクスは、製品の初期バージョンですでに登場する。同時に、機能の正確性とターゲットとなるユーザーの関心度の検証も始まる。

メタメカニクスは通常、ゲームに後から追加され、製品版のプロダクション段階の後半にテストが行われる。ゲームセッションの平均時間、招待された友人の数、平均支払額、最も人気のあるゲーム製品などへの影響の評価は、ベータテストの段階で行われる。同時に、A/B テストは、最も効果的なメカニクスを決定するためによく使用される。

2.1.4　クライアント、サーバー、クライアントサーバーメカニクスのテストの違い

クライアントサイド、サーバーサイド、クライアントとサーバーのメカニクスをテストする必要性は、ゲーム自体のアーキテクチャによって異なる。これらのメカニクスのタイプは、それぞれ実装の詳細によって、テストに異なるアプローチを必要とする。

クライアントメカニクスはプレイヤー自身のデバイス上で処理されるため、ユーザーはクライアントに保存されているすべてのデータにアクセスできる。ゲームによっては、完全にクライアント上で動作し、サーバーをまったく使用しないものもある。このようなゲームは通常シングルプレイで、インターネット接続を必要としない。

ゲームクライアントには、プレイヤーのデバイス上で実行されるコードと、すべてのパラメータ、ダメージ計算式、ゲームレベル、キャラクターモデルなどが含まれる。そのため、プレイヤーは必要に応じて、ゲームクライアント内のすべてのデータを変更することができる。クライアントを修正することで、ユーザーは、例えば、技のスピード、キャラクターの体力量、タスク完了時の報酬の値などを増やすことができる。

このようなクライアントの変更は、プレイヤーにゲーム内での優位性を与えるかもしれないが、シングルプレイヤーゲームでは、これは一般的に大きな欠陥とはみなされない。例えば、シングルプレイヤーゲームでは、コンピューター相手に剣で戦う仕組みが実装されるかもしれない。この場合、武器のパラメータはサーバー上で妥当性確認をされない。従って、特定のプレイヤーがゲームクライアントに変更を加え、剣のダメージを増やしても、それは彼個人のゲームプレイを変えるだけである。自分のデバイスで同じゲームをプレイしている他の独立したプレイヤーの利益には影響しない。

クライアントメカニクスはマルチプレイヤーゲームにも存在する可能性があるが、それでも他のプレイヤーに影響を与えることはない。例えば、レベルマップを見たり、キャラクターのインベントリを開いたりといったメカニクスは、クライアントサイドのみで処理され、サーバーが関与しない機能テストによって検証される。

クライアントメカニクスのテストは、通常、ブラックボックステストを使って実施される。これを行うとき、テスト担当者はアプリケーションのユーザインターフェースをチェックする必要がある場合がある。このケースでは、実際の結果を得るためにゲームをテストするとき、ユーザインターフェースそのものを使うことができる。

サーバーメカニクスはサーバーサイドのみで実装される。この機能により、メカニクスのロジックはユーザーの介入から保護される。プレイヤーはリモートサーバー上で実行されるプロセスに直接影響を与えることができないため、開発者はゲームロジックの計算の最も重要な部分をそこに転送する。

サーバーメカニクスのテストは、多人数参加型オンラインゲームにおいて最も重要である。例えば、プレイヤーがプレイヤー対プレイヤー（PvP）モードでチーム戦を始める場合、ゲームレーティングとキャラクターレベルがほぼ等しいサーバーで、味方と対戦相手の選定が行われる。そうでないと、参加者全員にとって面白くない、居心地の悪い競技になってしまうかもしれない。

サーバーに実装された機能をテストするアプローチは、クライアントメカニクスのテストとは異なる。この場合、テスト担当者は一般的にユーザインターフェースを必要としない。テストはサーバーのコンソールで、あるいは特別なツールを使って行われる。テスト担当者は、ほとんどの時間をログの解析やデータベースとのやりとりに費やす。

サーバーメカニクスの最も重要なテストタイプは、機能テスト、性能テスト、セキュリティテストである。

クライアントサーバーメカニクスは、クライアント側とサーバー側の両方が関与し、先に述べた 2 つのタイプのメカニクスの特徴を組み合わせたものである。ロジックの一部はプレイヤーのデバイス上で処理され、もう一方はリモートサーバー上で処理される。このようなメカニクスの各部分は常に互いに「通信」し、相互にデータを送信する。マルチプレイヤーゲームでは、サーバーがクライアントからデータを受信し、アクションの結果を、そのアクションの影響を受けるすべてのサーバープレイヤーに送信する。

サーバーに到着したデータは、必ず妥当であるかの確認を受ける。例えば、ユーザーがゲーム内ストアで何かを購入したい場合、サーバーはプレイヤーの進行状況に応じて、購入可能なゲーム内通貨が十分な量あるかどうかをチェックしなければならない。プレイヤーがゲームを有利に進めるために端末のクライアント部分を修正し、偽のパラメータをサーバーに送信しようとしても、サーバーは正しいデータに置換する。

テスト担当者がクライアントサーバーのメカニクスをテストする場合、最大数の異なるメカニクスをカバーするように、1 つ以上のテストケースを作成する。

このようなテストの例としては、クライアントがゲームサーバーにログインし、コンピューターがコントロールする対戦相手を倒す、というようなものがある。このテストを構成する必要なアクションの数は少ないにもかかわらず、クライアントとサーバーの相互作用のほとんどすべてのメカニクスがここで使われている：

- クライアントから承認担当サーバーにリクエストを送る
- 承認サーバーからデータベースへのリクエストを行い、承認を実行し、以前のビデオゲームステートを復元する
- サーバーにすでにロードされているマップを選択するか、サーバー上で新しいゲームメカニクスを作成する
- マップのあるゲームメカニクスサーバーにクライアントを接続する
- サーバーがオブジェクトを作成し、それをユーザーに表示するコマンドをクライアントに与える間、ゲームキャラクターとキャラクターの周辺にあるゲームオブジェクトのビューを提供する
- 新しいゲームオブジェクトを表示しながら、キャラクターをマップ上で移動させる
- コンピューターの敵とその攻撃を発見し、クライアントに表示する
- サーバーから「攻撃」コマンドを受信する
- キャラクターと移動体オブジェクト（Mob）が攻撃可能な距離にあり、キャラクターと Mob の間に攻撃を妨げるオブジェクトがないことを確認することで、攻撃が可能であることを検証する
- 攻撃が可能である場合、サーバーから Mob に被害を受けた旨のメッセージを送信する
- サーバー上の Mob が受けたダメージ量を計算し、Mob の体力のパラメータの値を減少させる
- Mob がプレイヤーに与えたダメージを表示する

2.1.5 ゲームメカニクスの欠陥の例と発生原因

ゲームメカニクスには、ゲームオブジェクトへの影響と、これらの影響の結果を示すフィードバックが含まれる。これらを組み合わせることで、適切なゲームダイナミクスとともに、ゲーム独自のキャラクターが生み出される。多くの場合、ゲームにはさまざまな影響やフィードバックの要素が使われる。

ゲームメカニクスに関連する欠陥には次のような種類がある：

- メカニクスにおける機能的欠陥
- ゲームの適切度認識性の欠陥

- 特定のゲーム環境で現れるメカニクスの有効性の欠陥

プレイヤーがゲームメカニクスの欠陥について語るとき、ゲームの機能的欠陥が最も多く言及される。例えば、武器がリロードされない、戦利品が集まらない、双眼鏡を使っても画像が近づかない、などである。このような欠陥は通常、開発者によるゲームコードのエラーから発生する。このような不具合は非常に目立ち、発見するのも修正するのも簡単である。

ゲームの適切度認識性の欠陥は、特定のメカニクスを使用している際に、ゲームから応答を得ることに関連している。プレイヤーがゲームオーバーの理由を理解できない状況は正しくない。そのため、ゲームのメカニクスには視覚効果や音響効果が伴うことが多い。これは爆発の視覚効果、タイムカウンターの音、あるいはテキストメッセージの場合もある。メカニクスを使用する際の反応の有無そのものがテストされ、その有無の正当性が評価される。エフェクトの動作と正しさは、他のテストタイプ（グラフィックやサウンドのテスト）でテストされる。

三番目の欠陥は、メカニクスの有効性と範囲に関するものである。メカニクスはそれ単体では効果的であっても、ゲームプレイの中で機能しなくなる可能性がある。そのような事態を避けるために、メカニクスは他のメカニクスやテスト対象、その他ゲームレベルで必要なコンテンツと組み合わせてテストされる。そのため、統合テストは、使用性テストとともに実施され、ゲームプレイにおけるメカニクスの有効性をチェックする。

例えば、ゲームデザイナーは、異なる行動や特徴を持つ2人の対戦相手をゲームに加えるかもしれない。1 人目は素早くて飛び跳ねるが、与えるダメージは小さく、2 人目は遅くて扱いにくいが、正確で強い打撃を与える。特徴という点では、どちらの対戦相手も同等に見え、またそれぞれに長所と短所がある。それぞれ独自の行動アルゴリズムを持つことができ、理論的にはプレイヤーに与える危険度はほぼ同じになるはずである。しかし、ゲームレベルの大部分が、プレイヤーキャラクターが登ることのできるさまざまな丘で構成されている場合、状況は一変する。遅い敵が近づけない高さにいるため、プレイヤーは安全に敵を攻撃できる。このような場合、丘を利用し、高速で移動し、高くジャンプする別の敵に立ち向かうのは難しい。この問題を解決するには、メカニクス自体の変更と、レベル上のオブジェクトの位置の変更の両方が考えられる。

このようなテストの複雑度は、主にそのような状況や、その中でメカニクスがどのように働くかを予測することの難しさと関連している。そのため、メカニクスとテスト環境対象との相互作用の問題点の探索は、しばしばアドホックテストの一環として行われる。ベータテストの段階で、大人数のプレイヤーを対象に行われることが多い。

2.2 ゲームメカニクスのテストアプローチ

2.2.1 ゲーム製品のソフトウェア開発ライフサイクルを通じたゲームメカニクスのテスト手順とアプローチ

ソフトウェア開発のさまざまな段階において、ゲームにおけるメカニクスのテストは、多数のテストアプローチを用いて実施することができる。

ゲームのプロトタイプを作成する段階では、コアメカニクスのみが実装されている。ここで、テスト担当者の主な仕事は、以下の特徴に対してゲームメカニクスをテストすることである：

- 機能正確性
- プレイヤーの関心度
- プレイヤーにとっての魅力

すべてのメカニクスとその運用性が、ゲームデザインドキュメントに完全かつ明確に記述されていることを確認するためにレビューする。ユーザーとメカニクスの相互作用について、許容されるすべての選択肢を検討し、各メカニクスがゲームプレイ全体に与える影響を決定する。

プロダクション段階では、実装されたメカニクスは機能テストにかけられ、記載された要件を満たしているか確認する。原則として、ここでのテストは一連のテスト条件とチェックリストを用いて行われる。テストの順序は、メカニクスのすべての側面を最大限にカバーすることを目指すべきである。探索的テストとアドホックテストは、異なるメカニクスの相互作用のテストに効果的である。このことは、特に大規模なマルチプレイヤーゲームにおいて顕著であり、メカニクスとゲームのさまざまなコンポーネントとの相互作用の方法は、テスト条件に反映しきれないほど多くなる。

非機能テストの一環として、ゲームの全体的な性能に対するメカニクスの有効性をテストする。資源効率性（RAM、CPU など）と実際の結果（ゲームクライアントのフリーズ発生、フレームレートの大幅な低下など）がテストされる。各種コンポーネントやソフトウェアの互換性、特にウイルス対策プログラムとの互換性をテストする。

ゲームがサーバーの存在を要件とする場合、サーバーの性能テストも行われる。性能テストは、パフォーマンスに影響する可能性のあるメカニクスが実装されるプロダクション段階で計画され、実行される。性能テストの計画と実行の詳細は、[ISTQB_FL_PT]に記述されている。

多数のプレイヤーによるベータテストを実施する際には、メタメカニクスに細心の注意が払われる。プレイヤーにとっての利便性とわかりやすさが評価される。非ゲームソフトウェアのテストと同様、このテストの主な目的は、明らかな機能的欠陥を発見することではなく、理想的には前の段階で特定されるべきものであるが、エンドプレイヤーからのフィードバックを得ることである。しかし、例えば大規模なマルチプレイヤーゲームなどでは、エンドプレイヤーも初期のアルファテストに参加し、フィードバックを提供することができる。

ゲームが市場にリリースされた後、テスト担当者の仕事は、プレイヤーから報告されたメカニクスの欠陥に対処し、チェックすることであり、ゲームに追加された新しいメカニクスをテストすることである。

2.2.2　ゲームメカニクスのテストの重要性

ゲームの骨格となるのはゲームメカニクスである。プレイヤーを飽きさせないために必要な、プレイヤーの体験の基礎を形成する。ゲームメカニクスはあらゆるゲームの本質であるため、その正しい運用性をテストすることはテスト担当者にとって最優先事項であり、そのようなテスト中に発見された欠陥は通常、最も重大なものとなる。

コアメカニクスの欠陥、例えば、キャラクターのジャンプが動作しないなど、コアメカニクスに欠陥があると、特定の障害を乗り越えることができず、結果としてゲームをクリアすることができない。

しかし、そのメカニクスがゲーム内でめったに遭遇しないものであったり、ビデオゲームの合格に不必要な活動（例えば、キャラクターが馬に乗る能力）に関連するものであったりしても、その誤った運用性は、ゲーム全体に対する認識に悪影響を及ぼす。

2.2.3　ゲームメカニクスのレビューの重要性

ゲーム制作において重要な段階は、特にゲームメカニクスを説明するゲームドキュメントの作成である。

開発段階の早い段階でゲームメカニクスの仕様をレビューすることは、後々の多くの問題を回避することに役立つ。このような問題は、開発プロセスそのものに関連することもあれば、最終製品に実装されたゲームメカニクスに対するプレイヤーの反応に関連することもある。

レビュー中、テスト担当者は以下の品質特性[ISO 25010]に注意を払うべきである：

- 記述の完全性
- 現実への準拠

- ドキュメントの構成とナビゲーションのしやすさ

レビューアが防げるゲーム開発プロセスの問題：

- 開発者がメカニクスの本質を正しく理解しておらず、メカニクスの明確かつ詳細で曖昧さのない説明が不十分であったために、必要な労力の評価を誤ること
- 新しいメカニクスと既存のメカニクスの非互換性
- 特定のプロジェクトにメカニクスを導入できないこと

レビューによって防ぐことができる、メカニクスに対するプレイヤーの認識に関する問題：

- 特定のゲームにおけるメカニクスの一般的な不適切さ
- 面白くないメカニクス
- バランスの悪いメカニクス
- ゲームの中で稀に使われるメカニクス

2.2.4 セッション再開後およびユーザーが非アクティブなときのビデオゲームステートのテスト

ビデオゲームステート

ビデオゲームステートは、特定の時点におけるゲーム内のすべてのオブジェクトを記述するすべてのパラメータと変数の値として理解される。

ゲームが複雑で、プレイヤーに提供するアクションの実現性が高ければ高いほど、各オブジェクトのパラメータは増える。

すべてのパラメータは、条件に応じて表示と非表示に分けることができる。

表示されるパラメータの値に関する情報は、プレイヤーに明示的に提供される。これには以下が含まれる：

- 完了したタスクに対するユーザーのゲーム内経験値
- 選択した武器の特徴
- インベントリ内のアイテムの最大数
- ゲーム内ストアの商品の値段

隠しパラメータは、開発者がプレイヤーを意識することなく、ゲームの進行に影響を与えるために使用することができる。例えば、プレイヤーに知られている攻撃ゲージと防御ゲージに加えて、キャラクターは、プレイヤーから隠されていて、ショットがターゲットにヒットする確率を決定する精度パラメータを持っているかもしれない。多くの場合、プレイヤーはこのようなパラメータの存在や、パラメータが具体的に何に影響するのかを知らない。

隠されたパラメータは、ゲームのさまざまな側面に影響を与える可能性がある：

- ドロップ率、つまり報酬としてランダムなアイテムを入手できる確率
- さまざまな種類の環境での発射速度や機器の移動速度
- プレイヤーの選択がゲームのプロットの展開に与える影響

一連の隠されたパラメータと明示的なパラメータにより、プレイヤーやその他のオブジェクトが操作するゲームキャラクターは独自に定義される。

ビデオゲームステートのテスト目的

ゲームパラメータをテストする際の重要なポイントは、ゲームが本当にパラメータの実際の値を使用しているかどうかをテストすることである。

明示的なパラメータの場合、テストは表示値と実際に使用された値を視覚的に比較することで行う。例えば、攻撃パラメータが 1 のゲームキャラクターが敵に 10 単位のダメージを与えた場合、プレイヤーが攻撃パラメータを 2 に上げると、同じ敵に与えるダメージ量も増え、例えば 20 単位になるはずである。

このようなテストが特に重要なのは、新しいダメージ量に関する情報が、例えば「攻撃パラメータ 1 単位につき 10 ダメージ追加」のようなゲームヒントの形でプレイヤーが利用できる場合である。この場合、プレイヤーはヒントに示された値と実際のダメージを比較することができる。

通常のプレイヤーは、パラメータの具体的な値を考慮することなく、ダメージの変化の妥当性と一貫性だけに注目するのが普通である。そのようなプレイヤーは、間違った点数のダメージを与えることを欠陥とは考えないかもしれないが、ダメージが変化しなかったり減少したりすれば、それは確実に注目される。

テストのプロフェッショナルは、ドキュメントに明記されているパラメータの変化を計算するための公式や原則に従う。この情報がドキュメントから得られない場合、テスト担当者はプレイヤーと同様、常識に基づいて相対的に評価する。

隠しパラメータをテストするために、テスト担当者はゲームデータベースにアクセスする必要がある場合がある。その助けを借りて、テスト担当者はパラメータの実際の値と予想される値を比較することができる。

また、ゲーム中にユーザーがキャラクターの明示的なパラメータを変更できることを前提とした場合、パラメータを変更する可能性とその表示の正確さがテストされる。

ゲームステートのテストは、ゲームのセーブとロードに密接に関係している。

セーブとロード

多くのゲームは、1 回のセッションで完了することは不可能であり、原則として無限のゲームプレイが可能であるようにする。そのため、ゲームではプレイヤーが毎回最初から開始する必要がないようにビデオゲームステートの「セーブ＆ロード」機能がよく使われる。

セーブを使用する他の理由としては、ユーザーにゲーム世界の探索を促し、ゲームによって提供される問題を解決するためのさまざまな方法を見つけて、さらにはゲームの複雑度を管理することである。プレイヤーが困難な障害物の前で生き残る機会があれば、間違った行動で進行が完全に無効になるよりも、エラーの代償は小さくなる。

セーブとは、ハードドライブまたはクラウドにセーブされるゲームの進行状況に関する情報のことである。セーブは、ゲームの現在の状態に関する情報を含むファイルとして表示される。このようなファイルのサイズはゲーム自体の数倍小さく、通常は以下の情報が含まれている：

- ステート情報をセーブする必要があるオブジェクトのリスト
- 各オブジェクトの変更パラメータのリスト
- ちょうどそのときにゲームで何が起きているのか判断ができる、各パラメータの特定のコード

ほとんどすべての変数がオブジェクトとして機能する。その状態はゲームシステムによって操作されるため、記憶される必要がある。変数オブジェクトには、例えばゲームキャラクターのヘルスレベル、学習したアビリティの数、ゲームマップ上のキャラクターの座標などがある。

セーブタイプとテストエリア

セーブ方法、セーブファイル形式、セーブされるパラメータのリストは、ゲームによって異なる場合がある。ビデオゲームの歴史を通じて、進行状況をセーブするために多くのアプローチが考案されてきた。最も一般的に使用されているものを以降で説明する。

セーブのためのチェックポイントの使用

セーブは開発者が指定したタイミングで自動的に行われる。次のチェックポイントに到達するごとに、プレイヤーの現在の進行状況に関する情報が上書きされる。チェックポイントにセーブされたビデオゲームのステートは、現在のゲームセッションの間だけセーブされ、ゲームが終了すると削除される。

場合によっては、ユーザーがチェックポイントに到達したことが明示的にゲーム上に表示されず、進行状況がセーブされることがある。セーブとロードの正しさをテストするために、テスト担当者は、すべての既存のチェックポイントのリストと正確な位置を提供するドキュメントが必要である。他のセーブ方法を使用する場合、ドキュメントにこの情報がなかったり、詳細が記載されていなかったりする場合がある。

固定セーブの使用

ゲームの特定の場所に、開発者はプレイヤーが必要に応じて現在の進行状況を手動でセーブできる特別なポイントを配置する。これらのポイントは通常、いくつかの限られたビデオゲームオブジェクトに囲まれており、大きなセーブファイルの作成を避けることができる。

この方法での検証範囲は、各ポイントでセーブできることと、セーブした各状態を繰り返しセーブ・ロードできることである。

オートセーブ

これはチェックポイントと固定セーブの組み合わせである。プレイヤーの進行状況は、ゲームプレイ中、特定のポイントで自動的にセーブされる。例えば、多くのゲームには終了時のオートセーブ機能がある。ゲームを始めるとき、ユーザーは新しいゲームを始めるか、以前にセーブしたゲームをロードすることができる。

この場合のテスト担当者の仕事は、すべてのポイントでセーブが行われたことと、到達しているビデオゲームのステートをロードできるかどうか確認することである。

手動またはフリーセーブ

セーブは、ゲームメニューの特別な項目を使用して、ゲーム中にいつでも行うことができる。高速セーブは、セーブとロードが 1 つのキーまたは 1 つのクリックで行われる場合に実装される場合がある。

オートセーブと同様に、手動セーブは、ビデオゲームのステートに関する情報を含む特別なファイルを作成する。このケースのテストには、ファイルからのロード情報の正しさ、ファイルの正しい名前、オペレーティングシステム構造内のファイルの場所、ゲームの新しいバージョンでの正しい操作、さらには、別のコンピューター上などでセーブファイルを使用する可能性の確認が含まれる。

以下の説明に当てはまるゲームでは、必要なゲームレベルが正しくロードされることをテストすれば十分である：

- ゲームでは、唯一の正しいルートとクリア方法が想定されている（例えば、迷宮から脱出する正しい方法は 1 つだけ）
- プレイヤーが操作するゲームキャラクターは、レベルごとに変化しない（例えば、衝撃力、ジャンプ範囲、レベルを通過する移動速度など、レベル通過のスピードを上げるような特性は増加しない）

以下の説明に当てはまるゲームでは、セーブにはゲームセッション、レベル、キャラクターなどに関する固有の情報が含まれる：

- ゲームキャラクターは、多くのパラメータ（移動速度、ジャンプの範囲や高さ、衝撃力、インベントリ、3D モデルの外観、およびその他の特性など）を持つことができる
- 上記のパラメータは、プレイヤーのゲーム内での行動や決定によって変化することがある（例えば、プレイヤーが迷宮でアーティファクトを発見し、移動速度が上がるなど）

このようなゲームの場合、テスト担当者は、ゲームの要件レベルや固有情報だけでなく、各プレイヤーや各ゲームキャラクターの固有情報を正しくロードしているかどうかをテストする必要がある。セーブをテストする場合、テスト担当者は以下のことをテストする必要がある：

- プレイヤーは前のゲームセッションを終了したときのゲームレベルになる
- プレイアブルキャラクターがアーティファクトを持っている
- このアーティファクトによって、ゲームキャラクターのスピードパラメータは確かに上昇する

3. グラフィックステスト - 165 分 (K3)

テストキーワード

プレイテスト

ビデオゲームに特化したキーワード

3D モデル、アニメーション、コリジョン、ビデオゲームレベル、当たり判定、レベルオブディティール（LoD）、マッピング、リギング、シーンライティング、スキニング、テクスチャー、視覚効果（VFX）

第 3 章の学習目標

3.1 ゲームグラフィックスの原理と概念
GaMe-3.1.1 (K2) ゲームプロダクトのグラフィックコンテンツの特徴を説明する。
GaMe-3.1.2 (K2) グラフィックコンテンツの欠陥の種類を分類する。

3.2 ゲームプロダクトにおけるグラフィックのテストアプローチ
GaMe-3.2.1 (K2) アーティスティックテストの主なアプローチを要約する。
GaMe-3.2.2 (K2) テクニカルテストの主なアプローチを要約する。
GaMe-3.2.3 (K2) ゲームプレイテストの主なアプローチを要約する。

3.3 グラフィックステストの実行
GaMe-3.3.1 (K3) グラフィックステストの基本的なアプローチを適用する。
GaMe-3.3.2 (K2) グラフィックスの歴史的な妥当性をテストすることの重要性を説明する。

3.4 グラフィックステストに対応するツール
GaMe-3.4.1 (K2) グラフィックステストのツールの使用法を要約する。

3.1 ゲームグラフィックスの原理と概念

3.1.1 ゲームプロダクトのグラフィックコンテンツの特徴

モバイルゲーム、大規模な高予算のゲーム、どのようなゲーム製品でもユーザーがゲームの進行を視覚化できるように、さまざまなグラフィックス要素が多数仕込まれている。

ゲームのグラフィックオブジェクトを作成するプロセスでは、さまざまな専門家が特定の役割を持つ：

役割	責任
アーティスト	ゲームのためのさまざまなグラフィックコンテンツを作成する。
3D モデラー	ゲームオブジェクトの 3D モデルを作成する。
テクスチャーアーティスト	ゲームオブジェクトのテクスチャーを作成する。
アニメーションスペシャリスト	ゲームオブジェクトのアニメーションを取り扱う。
テクニカルアーティスト	テクニカルテストを実施する。

役割	責任
テスト担当者	ビデオゲームエンジンのグラフィックオブジェクトのテストを行う。

ゲームプロダクトのテスト担当者は、プロダクト予算やその中のグラフィックコンテンツの量に関係なく、それぞれのゲームに部分的に、完全に、または別々に存在するさまざまな領域を考慮してテストする必要がある。

以降でテストするゲームプロダクトの主な領域をまとめる。

レベル（ゲームマップ）

レベルとは、ゲームの仮想世界の独立した領域のことで、通常は特定の場所、例えば建物や都市を表す。この用語は、初期のテーブルトークロールプレイングゲーム（RPG）に由来し、ダンジョン（つまり、ゲームの大部分が行われる環境）の階層を指している。そのゲームではプレイヤーはダンジョンの最深部（レベル 1）からスタートし、最後のレベル（例えばレベル 100）に到達するまで、次第に難しくなるすべてのレベルを通過して地上に到達し、ゲームに勝利しなければならなかった。今では、ほとんどすべてのビデオゲームに、何らかの形でレベルが存在する。

モデル

モデルとは、コンピュータグラフィックスにおけるオブジェクトを指す。画像を作成する方法によって、グラフィックスは以下のように分類される：

- 2D グラフィックス
- 3D グラフィックス
- コンピュータ生成画像（CGI）

2D グラフィックス

2D コンピュータグラフィックスは、グラフィック情報の表現形式とそれに続く画像処理アルゴリズムによって分類される。通常、2D コンピュータグラフィックスはベクトルとラスターに分けられるが、フラクタルタイプの画像表現もある。

3D グラフィックス

3D グラフィックスは、3 次元空間のオブジェクトを操作する。3D コンピュータグラフィックスは、映画やコンピューターゲームで広く使われている。

3D グラフィックスは以下のようなものがある：

- ポリゴン
- ボクセル

ポリゴングラフィックスとは、3D コンピュータグラフィックスやボリューメトリックモデリングにおいて、多面体オブジェクトの形状を定義する頂点、辺、面の集合体である。面は通常、三角形、四角形、または他の単純な凸多角形で、レンダリングを簡素化する。

ボクセルとは、6 つの矩形ポリゴンの集合に対応する仮想要素である。仮想世界のすべて（ピクセル、ポリゴン、ボクセル）は、物理的なスクリーンのピクセルに投影されなければならない。ボクセルグラフィックスはラスターグラフィックスと類似している。オブジェクトは 3D 形状の集合で構成され、多くの場合は立方体である。

コンピュータ生成画像（CGI） グラフィックス

CGI とは、コンピューターが計算に基づいて作成した 3D 画像のことで、ビジュアル・アート、印刷、映画の特殊効果、テレビ、シミュレーターなどで使用される。動画は CGI グラフィックスの狭い領域であるコンピューターアニメーションで作成される。

モニターは列と行で構成されるマトリクスであるため、モニター上の画像はその平面性により、すべてラスターとなる。3D グラフィックスは想像上の存在でしかなく、見る人の創造によって作り出され、モニター上では 3D モデルが投影されたものとして見える。従って、グラフィックスの視覚化はラスターとベクターしかありえず、レンダリング方法もラスター（＝ピクセルの集合）でしかない。画像の定義方法は、このピクセルの数に依存する。

テクスチャー

テクスチャーとは、ポリゴンモデルの表面に色やレリーフの錯覚を与えるために適用されるビットマップのことである。大まかには、テクスチャーは、彫刻的なイメージの表面上のパターンと考えることができる。テクスチャーを使用することで、ポリゴンで作成するにはリソースを過剰に消費するような小さな表面オブジェクトを再現することが可能になる。例えば、皮膚の傷跡、衣服の折り目、壁や土の表面にある小さな石やその他のオブジェクトなどである。

テクスチャーサーフェスの品質は、最小テクスチャーあたりのピクセル数を表す単位であるテクセルによって決定される。

テクスチャーのテストにおける主な側面は、目に見えるオブジェクト上の存在、正しい表示、均一性をテストすることである。ゲームのテクスチャーが高解像度であれば、低品質なテクスチャーは存在してはいけない。

コリジョン

コリジョンは、それぞれのオブジェクト間で通過、衝突の特性を担う。

コンピューターゲーム、特にコンソールゲームでは、限られたハードウェアリソースと非常に限られたゲームプレイ時間の間で、多くのタスクを分配しなければならない。このような制限や、比較的未発達で不正確な衝突検出アルゴリズムの使用にもかかわらず、ゲーム開発者は、ゲームオブジェクト同士の相互作用を視覚的に信じられ、比較的現実的な方法で表示する方法を作成することができた。

ほとんどのコンピューターゲームでは、衝突や侵入を避ける必要がある主なオブジェクトは、レベルの風景や環境となる。これらは、山、木、建物、フェンスなど、静的で非インタラクティブかつ破壊不可能な構造物である。この場合、キャラクターは 1 点のみで表現され、バイナリ空間分割法（ユークリッド空間を凸集合と超平面に再帰的に分割する方法）が使用される。その結果、オブジェクトは、観察者からの距離順に視覚オブジェクトを並べ替えたり、衝突を検出したりするなど、3D コンピュータグラフィックスに対する運用性を効率的に実行するために使用されるデータ構造の形で提示される。これは、キャラクターを表す点が環境（地形）にあるかどうかをテストするための、実行可能でシンプルかつ効果的な方法を提供する。キャラクターと他の動的オブジェクトの衝突は、別々に考慮され、処理される。コリジョンの詳細については 3.1.2 を参照。

アニメーション

モデルがアニメーション化される前に、モデルの中に「スケルトン」が作成される。このプロセスをリギングと呼ぶ。生き物のモデルや、アニメーションを想定したすべてのオブジェクトは、仮想の骨格を持つことができる。例えば、主人公のマント。モデルの骨格は互いに依存しており、例えば手が動くと手のひらの骨格も動く。後続のアニメーションはすべて、リギングとスキニングに依存する。アニメーションはオブジェクトの移動や形状の変化といった動く画像の錯覚を作り出す技術である。

モーフィングでは、高い頻度（手描きアニメーションの場合、1 秒間に 12 フレーム、コンピューターアニメーションでは 1 秒間に 30 フレーム）で入れ替えられる一連の静止画像（フレーム）が使用される。

エフェクト

視覚効果は、大きく 2 種類の作業に分けられる：

- ゲームプレイ効果（相互作用効果）
- 自然効果（環境的効果）

適用される原則は、特定のプロジェクトによって異なる。

他のジャンルの中でも、格闘ゲームや RPG ではゲームプレイ効果がより重要になる。例えば、自分のプレイアブルキャラクターが他のプレイヤーのプレイアブルキャラクターを押すとき、それぞれに正しいインタラクションアニメーションを再生する必要がある。3D シューティングゲームのような他のゲームジャンルもあり、特にリアルなものでは、ゲームプレイ効果と同じくらい重要な自然効果もある。自然効果の例としては、滝、霧、雨などがある。

シーンライティング

シーンライティングは、プレイヤーがシーンを見るために必要である。光は感情に影響を与える。絵と感情的な反応との結びつきは、キャラクター、物語、サウンド、ゲームメカニクスに役立つもう 1 つの強力なツールとなる。この場合、光と表面の相互作用によって、明るさ、色、コントラスト、影、その他の効果性に影響を与えることができる。

構造的な特徴により、人間の目は 110°以内で物体を立体的に認識し、さらに小さな範囲でフルカラーを認識する。中心視（上記の範囲内）は人間が最初に使うものなので、デザイナーが意図したように、プレイヤーが確実に見るべき重要な要素を得るべきである。周辺視野はコンテキストを提供し、中心視野を強化する。ビデオゲームでは、周辺視野に入る要素が必要なコンテキストを提供しなかったり、中心視野にある要素と矛盾したりすると、デザイナーとプレイヤーの接続性が切れてしまう。

シーンライティングの正しい使用例を以下に示す。:

- 中央の視野に降り注ぐ光は、プレイヤーを誘導することができる
- 照明はフレームを変えることができる。例えば、懐中電灯を使うと、懐中電灯がプレイヤーの主な光源になる。視点が変わることで、照らされた部分のプレイヤーの視界がとらえられ、強いコントラストのために他のすべてが切り取られる
- ライティングは、誘導することもできるし、恐怖感などの雰囲気を作り出すこともできる。例えば、暗闇のどこかに恐ろしい敵が隠れていると、プレイヤーは常に緊張を強いられる
- 光の方向は、ゲームレベルで要素を発見しやすくすることも、より複雑にすることもできる
- 照明が不足したり過剰になったりすると、プレイヤーは特別なアイテムやアクションを使わざるを得なくなる

シーンライティングはゲームの美観を作り出し、ゲーム体験に影響を与えるため検証で最も重要な領域の1つである。そのため、視覚芸術、映画、建築で見られるシーンライティングの多くのテクニックが、仮想空間の美しさを補完し、プレイヤーの体験を向上させるために、コンピューターゲームに使用されている。ただし、ゲームは映画や劇場とは大きく異なり、その環境はダイナミックで予測不可能である。静的なシーンライティングに加え動的な光源が使用される。これにより、インタラクティブ性と適切なエモーションが加わる。

歴史的正確性

グラフィック要素、アニメーション効果的といったさまざまな要素は、全体的なプレイスタイルにマッチしていなければならない。テキスト、サウンド、オブジェクトの説明、モデルやロケーションは、歴史的なプロトタイプの説明と一致していなければならない。

豊富で複雑なグラフィックコンテンツは、コンピューターゲームとその他のソフトウェアとの大きな違いの 1 つとなる。これにより、テストへのアプローチも大きく変わる。ゲームのグラフィックをテストするためには、テスト担当者は物理学や光学の知識を持ち、カラーレンダリングの技法を理解し、歴史に関する知識を持っていなければならない。グラフィックスのテストは、最も重要なテスト活動の 1 つである。なぜなら、欠陥の大部分がここに集中し、ユーザーのゲームに対する認識やユーザエクスペリエンスに直接影響するからである。

3.1.2 グラフィックコンテンツの欠陥の種類

グラフィックステストにおける欠陥の一般的なカテゴリーは、視覚的欠陥である。グラフィックの欠陥には、画面上の画像崩れ、テクスチャーの欠如、画像の特定の領域の予期せぬトリミングなどが含まれる。

モバイルゲームでグラフィックやアニメーションを作成する場合、PC と同じエンジンが使用される。異なるのは、同じエンジンが特定のプラットフォーム、そこで使用されるハードウェア、およびその技術的適応性に適合していることである[Gregory18]。そのため、このようなゲームで発生する欠陥は似ている。

テクスチャーの欠如

テクスチャーのテストにおいて、最も頻繁に遭遇する欠陥には以下が含まれる：

- グラフィックオブジェクトのテクスチャーの欠如
- ゲームプレイ中のテクスチャーの損失
- 一時的なテクスチャーとスタブ

経験が示すように、最初の 2 つの欠陥グループは、グラフィックスプロセッサの性能不足または古いグラフィックスカードドライバーに関連することがほとんどとなる。これに対して、一時的なテクスチャーは、開発の欠陥によって現れることがある。

レベルオブディティール（LoD）

最新の 3D ゲームでは、カメラ（プレイヤー視点）からの距離が異なる多数のオブジェクトを同時にステージ上に表示することができる。システムの負荷を軽減するために、開発者はレベルオブディティール（LoD）技術を使用する。

開発者が新しいゲームオブジェクト（木、建物、車など）を作成するとき、ゲームにオブジェクトモデルのバリエーションを追加する。これには、ポリゴン数が少なくジオメトリが単純化されたモデル（ローポリモデル）と、ポリゴン数が多く詳細なモデル（ハイポリモデル）がある。カメラとの距離によって、同じオブジェクトでもポリゴン数の異なるモデルが使用される。カメラに近い位置では、最大画質を実現する要件がある場合、ハイポリモデルが使用される。カメラが遠ざかるにつれて、ポリゴン数の少ない詳細度の低いモデルに置換される。十分な距離が離れると、モデルはシルエットとしてのみ表示されるか、まったくレンダリングされなくなる。これにより、処理ポリゴン数を減らし、ゲームのパフォーマンスを向上させることができる。

例えば、水平線上の森であれば、低解像度のテクスチャーを使用し、レリーフや反射を使わずに、その色だけを表示することができる。プレイヤーが近づけば、木々は完全に詳細に表示される。

LoD 技術は、アニメーション、スケルトン、ボットの人工知能にも使用されており、自動化されたプログラムが人間のプレイヤーに代わり特定のゲームをプレイする。例えば、フレーム内に多数の敵が登場するシューティングゲームでは、ボットのディティールレベルが異なる。背景のボットは、テクスチャーのディティールが低く、簡略化されて描かれるが近くにいる敵は、精巧なアニメーションで丁寧に描かれ、プレイヤーと戦えるほど賢く見える。

開発者は、現在のフレームレート、画面上のオブジェクトの移動速度、同時に表示されるオブジェクトの総数に応じて LoD も変更する。

LoD の使用から生じる欠陥は、主にシステムの性能の低さに関連している。開発者は、オブジェクトのディテールの増加やモデルの置き換えが、プレイヤーに気づかれずにスムーズに行われるように努めているが、必ずしもそうとは限らない。脆弱なハードウェアで動作するゲームでは、ローポリモデルがハイポリモデルにすぐに置き換えられないことがある。その結果、オブジェクトに近づいたプレイヤーは、まず「粗い」モデルを目にし、それからより良いモデルに変わる。

コリジョン

コリジョンとは、ゲーム内のオブジェクトがサイズを獲得し、他のオブジェクトとの衝突に反応する方法となる。コリジョンメッシュまたはコライダーは、完成したオブジェクトモデルが環境や他のオブジェクトと相互作用するために作成される。これはオブジェクトの目に見えない簡略化された形で、他のオブジェクトとの衝突を計算するために結びつけられる。コライダーはオブジェクトの物理モデルと呼ぶことができる。プレイヤーからは見えないが、コライダーの衝突はゲームエンジンによって処理される[Buttfield19]。

コライダーの形状は、モデルのメッシュにほぼ一致する必要があるが、かなり大まかな近似で十分なこともよくある。建物、岩、クラッシュした車など、環境内のほとんどの静的オブジェクトには、開発者はローポリコライダーを使用する。これはゲームプレイでは区別がつかないが、すべての可視オブジェクトの処理効率性を向上させる。

特にプレイヤーがこのオブジェクトと直接インタラクションできる場合は、コライダーのサイズは可視モデルに対応しなければならない。

オブジェクトのコライダーがそのビジュアルモデルよりもはるかに小さい場合、キャラクターは "テクスチャーの中に落ちる" または "テクスチャーを通り抜ける" ことができる。厳密には、「テクスチャーの中に落ちる」という表現（例えば、キャラクターが他のオブジェクトの中を通り抜けられる場合）は正しくないが、プレイヤーから見てキャラクターが部分的に、または完全に他のオブジェクトに没入していることを完璧に表現している。

このような故障はプレイヤーに不正なゲーム上の優位性を与える可能性があるため、マルチプレイヤーゲームにおいて最も致命的である。例えば、石の中にいる戦車は、他のプレイヤーからはほとんど見えなくなる。同時に、そのような石は敵の銃撃から身を守ることはできない。そのため、テスト担当者は、プレイヤー対プレイヤー（PvP）モードのマップでは、このようなオブジェクトにさらに注意を払わなければならない。

とはいえ、ゲーム内のオブジェクトの中には、装飾的でコリジョンがまったくないものもある。例えば戦車は藪の中を簡単に走行でき、プレイヤーは小さな石やブリキ缶をいちいち回り込んだり飛び越えたりする必要はない。

ここではゲームの慣例が適用されている。同じように、長い武器を持ったキャラクターを狭い低い通路に送り込み、さまざまな方向から攻撃してくる敵を撃退することができる。現実とは異なり、キャラクターは好きな方向に自由に曲がることができ、曲がるときに槍が壁を通過することにも気づかない。さらに、キャラクターの長い髪が鎧の巨大な肩パッドを突き抜けてしまうこともある。しかし、低い柵があると、プレイヤーは突然登れなくなり、プレイヤーを困らせ、楽しみを減らしてしまう。

オブジェクトのコライダーのサイズが可視モデルよりも大きい場合、キャラクターが見えない壁に当たっていたり、小さな支柱の上で空中に立っていたりする状況が発生することがある。このような故障は、ゲームの予期せぬ高速通過によく使われる。いわゆるスピードランナーは、開発者が怠慢だったり、周囲のオブジェクトのコライダーのサイズを変更し忘れたりした場所を特に探している。これにより、プレイヤーはその場所のかなりの部分を迂回し、通過時間を短縮できることがある。

当たり判定

キャラクターやゲームオブジェクトが、例えば敵の弾丸や投射物からダメージを受けると仮定した場合、戦闘中にこのオブジェクトの表面上の各点の位置を計算し、命中したかどうかを判断する必要がある。

複雑な形状のオブジェクトの場合、これは非常にコストがかかり、必ずしも正当化されない。

このような計算を簡単にするために、当たり判定という概念が使われる。例えば、2D のプラットフォーマーや格闘ゲームでは、1 つまたは複数の長方形の形にすることができる。お互いの相対的な位置を計算するのは簡単である。

3D オブジェクトの場合、最も単純な当たり判定の形状は球体である。この場合、計算にはその中心と半径がわかれば十分である。こうすることで、いつでも、弾丸や他の物体がこの球の中にあるかどうかを

簡単に判断することができる。しかし、平行なパイプ状の当たり判定は、球に比べて空いたスペースが少なく、オブジェクトの中心点と、計算のための長さ、幅、高さの 3 つの寸法がわかれば十分であるため、より適切な形状であることがわかる。

より真正性を高めるために、オブジェクトを複数の当たり判定で表現することができる。モデルの各部分：ボディ（腕、脚、頭、尻尾、武器など）。

しかし、モデルのダメージを受ける部分とダメージを与える部分は必ずしも一致しない。そのため、ゲームによっては、同じオブジェクトに対して、ダメージを与える領域（武器、拳など）とダメージを受ける領域（頭、手足など）が定義されるように、当たり判定とダメージ判定が別々に割り当てられている。キャラクターがダメージを受けるかどうかを判断するために、武器の当たり判定とキャラクターの体のヒットボックスが重なっているかどうかがテストされる。

当たり判定を扱う際のテスト担当者の主な仕事は、目に見える衝撃の範囲が実際の攻撃結果と一致しているかどうかをテストすることである。

大規模なゲームであっても、自分の武器が敵に触れてもダメージを与えなかったことがプレイヤーにはっきりとわかる瞬間がある。逆に、当たり判定が大きすぎると、目視では当たっていなくても、敵の剣がキャラクターを傷つけることがある。

その他の欠陥

大きなプロジェクトの作業成果物では、モデル制作の各段階で何度も承認と確認が行われる。アーティスト、モデラー、テクスチャー担当者などがテストを行う。それでもテスト担当者は、制作過程で生じた見た目の欠陥を発見することができる。

歴史的原型との不整合

プレイヤーは、ファンタジーや SF ゲームでは、巨大な肩パッドが付いた非現実的な装甲や、ありえないような飛行機械の外観を受け入れるかもしれない。しかし、ゲームがリアルであると主張するのであれば、ある機種のプロペラの位置がまったく違っていたり、ある戦艦の砲の数がずっと多かったりすることを指摘する几帳面なプレイヤーが必ずいるはずである。

そのため、テスト担当者は、可能であればテスト対象の写真を少なくとも 2、3 枚は見て、実際にどのように見えるかを想像しておくとよい。

空中にぶら下がる物体

多数のスペシャリストが同時に作業できるゲーム環境では、オブジェクトの追加や削除を行う者もいれば、地形（マップ）のジオメトリを変更したり、タイル（地形に貼るテクスチャー）を修正したりする者もいる。こうした運用で、古い「サポート」が取り除かれ、既存のオブジェクトが地形に埋もれてしまったり、空中に浮いてしまったりする欠陥が発生する。

目に見えるテクスチャーのつなぎ目

このような欠陥は、大きな地図の地形や、複数のテクスチャーが混在している場所で発見されることが多い。例えば、 "継ぎ目" は、草の表面と砂や石の間の移行部分で見えるかもしれない。詳細は 3.1 節を参照。

物体の破壊

通常、モデルは、元の状態、損傷した状態、破壊された状態など、さまざまな状況に現実感を持たせるためのさまざまな状態がある。ゲームがオブジェクトのリアルな破壊を想定している場合、オブジェクトのために特別なクラッシュモデルが作成され、場合によっては独自のコリジョンモデルも作成される。ある瞬間に、元のモデルは破壊されたものに置換される。

ここで重要なのは、破壊の効果がモデルそのものに対応するようにすることである。破壊されたトラックは元のトラックと異なっていなければならないし、壊れたレンガ造りの建物の破片は木製であってはならない。

照明欠陥

照明の欠陥は以下のとおり：

- グローバルイルミネーションの欠陥
- 点光源の欠陥
- サーチライト光源の欠陥
- 面積を持つ光源の欠陥
- 指向性光源の欠陥
- 発光光源の欠陥
- 拡散光源の欠陥

現代のビデオゲームエンジンには、光学法則に完全に従ってさまざまなタイプの光源を作成するための非常に高度な機能が搭載されている。しかし、間違った照明設定や開発者の経験不足などにより、照明が理想的でないゲームもまだ存在する。視覚的な構成（光の位置、角度、色、視野、動き）を考慮することは、プレイヤーがゲーム環境をどのように認識するかに大きな影響を与え、プレイヤーを惹きつける適切な雰囲気を感情的に作り出す。デザイナーは、ビジュアルのインテグリティを作ることで、この点に注意する必要がある。[Lee16]、[Tavakkoli18]、[Romero19]。

アニメーションの欠陥

スキニングと呼ばれる欠陥は、スケルトンをモデルにスナップすることから生じる。モデルの骨格が多ければ多いほど、よりリアルなアニメーションが作成できる。主人公の髪の毛など、さまざまなオブジェクトをアニメーションさせることができる。

モデルの骨格は互いに依存している。従って、手が変位すると、手のひらの骨格も動く。その後のアニメーションはすべて、リギングやスキニングの際に、どれだけリアルに行われ、設定されたかに依存する。

もしデザイナーがエラーを起こせば、さまざまなアニメーションでキャラクターの腕や足が長くなったり、モデルの一部が他の要素から「切り離されたり」することになる。多くの場合、これらの故障は、他のモデルと衝突したときや、モデルのさまざまなアニメーションで現れる。

視覚効果（VFX）の故障

ゲームでは、爆発、火花、煙、消失、出現など、登場人物の行動や出来事、自然現象にまつわるさまざまな効果的演出が多用される。

アニメーションで作業する場合、視覚効果的は通常、骨格、ヘルパー（モデルを作成し、アニメートするために使用されるユーティリティオブジェクト）、シーン内のオブジェクトにアタッチされる。視覚効果の作成は、アニメーションチームと視覚効果アーティストが相互作用する反復プロセスとなる。視覚効果アーティストは、アニメーションの変更や追加を求められることがある。例えば

- 剣の軌跡をより滑らかにするためにキーフレームを増やす
- ヘルパーを回転させ、その回転を血しぶきの方向として使用する
- パーティクルメッシュの醜い外観を避けるために、カメラの切り替えを変更する

まず、このような有効性は、それを生み出す事象と同期していることが重要である。例えば、銃口からの炎や煙は、発射の瞬間や銃身から弾丸が出てくる瞬間と同期していなければならない。そうでなければ、イベントのリアリズムが侵害されてしまう。このような現象は伝統的に故障とみなされる。

VFX の故障を引き起こすもう 1 つの問題は、フレームレートの維持とエフェクトを正しく表示するためのハードウェアリソースを最適化するための技術的条件に準拠していないことである。どんなに大規模なエフェクトであっても、画面上でピクピクと動く砂煙や爆発が止まっているのをプレイヤーが喜んで見るとは思えない。そのため、ゲームの特殊効果には厳しい技術的制限が課せられており、その目的は表示フレーム数を一定に安定させることにある。

VFX をテストする場合、テスト担当者はゲーム機器のハードウェア機能が最適に使用されていること、そしてエフェクトがそのトリガーとなるイベントと同期していることを確認する必要がある。

モデル製造段階でできるだけ多くの欠陥を発見することが望ましい。例えば、モデルのポリゴン数の超過やテクスチャリングの欠陥は、モデル計算に使用されるハードウェアの負荷を大幅に増加させる。これは、必然的にビデオアダプタープロセッサの過負荷によるゲームの長い遅延につながる。

これをテストするためには、開発者は LoD や衝突モデルなどの側面について、あらかじめ決められた固定要件を受け取っていなければならない。

テスト担当者は、原則として、欠陥によって引き起こされる故障や不具合を含め、これらの欠陥の兆候を発見する。例えば、ポリゴン数が多すぎるコリジョンモデルを持つオブジェクトの場合、このオブジェクトとのインタラクション中のフレームレートが高い限り、欠陥は検出されない。

3.2 ゲームプロダクトにおけるグラフィックのテストアプローチ

3.2.1 アーティスティックテスト

グラフィックオブジェクトを作成する過程では、さまざまな段階で各レビューを通過する。デザインを作成する人もいれば、それをアプリケーションに落とし込む人もいる。その結果、アウトプットはデザインの作者が意図したものとはまったく異なるかもしれない。テスト担当者は、設計の標準適合性をテストすることができる。しかし、アーティストがもっと簡単に気づく欠陥もある。例えば、背景の透明度が不十分であったり、アプリケーションがスケーリング時に意図したとおりに動作しなかったりする。

逆に、設計者はすべてを予見することはできず、欠陥がすでに故障となった時点で、実行中のアプリケーションにおいて初めて欠陥に気づくはずである。従って、重大な欠陥の除去を支援するために、実装後にアプリケーションを設計者に渡してレビューしてもらうことは良い習慣となる。このプロセスにおいて、テスト担当者は必要な環境を準備し、テストアカウントを作成し、テスト後の欠陥について記述し、それらをテストするか、あるいは、レビューのために設計者に送ることができる。

コンテンツ量が多いプロジェクトでは、グラフィックステストに携わる専門家がいることが多い。彼らは主にアーティストで、モデル制作のプロセスを熟知している。彼らは、モデル制作の以下の各段階の確認を行う：

- ジオメトリの作成
- テクスチャーの作成
- コリジョンモデルの作成

この作業の目的は、ビデオゲームエンジンにエクスポートする前のモデルの品質を向上させることである。例えば、ジオメトリのポリゴネージ（ポリゴンの数）、その位置の正しさ、モデルの展開の正しさ、細長いポリゴンの存在、モデルの継ぎ目などの側面をテストする。

アーティスティックテストは、最も単純なジオメトリとテクスチャーの作成から始まり、エンジンにモデルをエクスポートしてマップに配置するまで、制作のさまざまな段階で実行できる大規模で複雑なタスクである。アーティスティックテストを行うために、テスト担当者は特別なツールやコンテンツエディターを必要としないことが多い。ゲームを実行しながら直接行うことができる。しかし、ツールやコンテンツエディターの可用性により、欠陥をより速く、より効率的に発見することができる。

アーティスティックテストは以下の役割によって実施される：

役割	アーティスティックテストの責任
アーティスト	オブジェクトをレビューするとき オブジェクトを見たり評価したりするとき

役割	アーティスティックテストの責任
3D モデラー	オブジェクトをレビューするとき
監督	オブジェクトを見たり評価したりするとき
テストエンジニア	モデルを最終的にエンジンにエクスポートした後
プレイヤー	プレイテストに参加するとき

3.2.2 テクニカルテスト

テクニカルテストには、グラフィックスの技術的パラメータに関連する一連のタスクが含まれる：

- 先に述べたようにモデル・ポリゴン数の制限の遵守
- テクスチャーフォーマット
- LoD スイッチング距離。利用可能なリソースと仕様を比較すれば十分である

テクニカルテスト自体は、テスト担当者とテクニカルアーティストによって行われる。以下のような手順で行われる：

グラフィックコンテンツリソースの未使用ファイルや一時ファイルの検索

アーティストと一緒に作業していると、後で使われないリソースが誤ってリポジトリに追加されることがある。例えば、開発者が未使用のものを含むすべてのリソースをリポジトリにロードしてしまうことがある。この場合、ゲームクライアントがハードディスクに占有する容量が増え、パフォーマンスが低下し、最終的にプレイヤーに配信されるパッチの総量が増える（例えば、個々のビデオゲームファイルの更新）。これを防ぐために、未使用リソースの検索が実行される。

クライアントのリソースに必要なすべてのコンテンツ・レコードがあるかどうかの検索

ゲームクライアントリソースでは、グラフィックコンテンツは他のリソースから独立して存在しない。テクスチャーへの参照はモデルに記述され、モデルへの参照はマップに記述される。このような記録にわずかな欠陥があっても、予期せぬ結果につながる可能性がある。

テクスチャーフォーマットのテスト

テクスチャーの種類ごとにテクニカルアーティストは独自の要件を設定する。これは、コンテンツのパフォーマンスとビジュアルコンポーネントのバランスを取る必要があるためである。テクスチャーフォーマットは、圧縮のタイプとそのサイズを意味する。

コンテンツのテスト時のテスト結果記録作業

クライアントのサブシステムの仕事は、あまり目立たないこともある。例えば、ポストエフェクトは、太陽を見るときのレンズのように、画像の上に重ねて表示される効果となる。画面上の重ね合わせに合わせて、ポストエフェクトごとに専用のテクスチャーが作成される。例えば、太陽をレンズで見た場合、テクスチャーがポストエフェクトに割り当てられているかどうかに関係なく、レンズと太陽の光線が見える。この場合、欠陥が存在するにもかかわらず、プレイヤーは欠陥に気づかないだけとなる。このような欠陥の発見事項を見つけるには、クライアントのログを調べる必要がある。クライアントのログには、リソースに存在しないテクスチャーをレンダーで使用しようとしたことが記載されているはずだからである。

テクスチャー数、モデル数に関する制限に準拠しているかの検証

各コンテンツにはそれぞれ制限があり、通常は特定のグループに対して一般的なものとなる。例えば、破壊不可能な複数階建ての建物の場合、15k ポリゴンまで使用できるが、平屋建ての建物の場合は 10k ポリゴンまでとなる。コンテンツの編集が継続的に行われ、リポジトリへのダウンロード数が多い場合、モデルを新たにエクスポートするたびに、ビデオ・プロセッサ・モデルの計算に必要なハードウェアリソースの不足に関連する状況を防ぐために、制限と要件の遵守をテストする必要がある。

テストはシンプルだが、テクニカルテストの重要性を理解することは不可欠である。グラフィックはシステム全体のパフォーマンスに大きな影響を与える。そのため、テクニカルテストには性能テストも含まれる。ゲームの性能とサイズが望ましいレベルに保たれていることを確認するためとなる。テクスチャーやモデルの数を増やすと、ビルドのサイズも大きくなる。このテクスチャーフォーマットの不一致はフレームレートの指標を悪化させ、クライアントが消費するメモリの増加につながる。テクニカルテストは、芸術的なコンテンツがプレイヤーに最大限の範囲と範囲で提供されることを確認する。

3.2.3 ゲームプレイテスト

ゲームプレイテストは、ゲームプレイに影響を与える要素を対象としたテストアプローチである。これには、コリジョンモデルとビジュアルモデルの標準適合性の評価と、ゲームプレイオブジェクトの設定がゲームモードの要件に適合していることの検証の両方が含まれる。コリジョンモデルとビジュアルモデルの適合性の評価と、ゲームプレイオブジェクトの設定がゲームモードの要件に適合していることの検証の両方が含まれる。

テスト対象のゲームプレイテストは以下のとおり：

- ヘルスポイント（HP）の適合性のテスト。これはロールプレイングゲームやコンピューターゲームにおける値で、オブジェクトを破壊するために与えられるダメージの最大量を決定する。HP はオブジェクトの材質、サイズ、種類を考慮して自動的に割り当てられるにもかかわらず、欠陥が発生する可能性がある。そのため、HP の目的はオブジェクトの特性に合わせて手動でテストされる
- コリジョンモデルがビジュアルモデルと一致するかどうかのテスト。前述したように、コリジョンモデルはビジュアルモデルに比べてかなり単純化されている。このため、プレイヤーが視覚的にはモデルの後ろに隠れていても、実際には敵によってダメージを受けることがあるという欠陥が生じる可能性がある

グラフィックのテストは複雑なプロセスであり、そのタイプは互いに重複している。前述のテストアプローチにより、ゲームのグラフィックコンテンツがさまざまな段階で徹底的にテストされ、グラフィックの欠陥が検出される。

3.3 グラフィックステストの実行

3.3.1 オブジェクト製作のさまざまな段階でのグラフィックステストの実行

グレーボックスの作成

これはグラフィック対象の簡略化されたモデルで、実際にはゲームプレイのテストに使われるモックアップである。この段階で発見された欠陥は、主にアーティストがこのモデルで作業する際の利便性に関するものである。

可視ジオメトリの作成

モデリングとマッピングの次の段階は、環境の物理的なオブジェクトに 3D 投影のマッピングと作成が含まれる。これは、ジオメトリと空間内の位置を考慮したものである。モデルの可視ジオメトリと UV 展開が作成され、3D オブジェクトの表面上の座標（X、Y、Z）とテクスチャー上の座標（U、V）の対応がチ

ェックされる。インスペクションは 3D グラフィックスエディターを用いて行われることが多いため、この段階での欠陥はアーティストや 3D モデラーによって検出されることもある。これらの欠陥には、原則として以下のようなものがある：

- ジオメトリの複製。ポリゴンの複製はパフォーマンスを著しく低下させる
- 過剰なディテール。例えば、プレイヤーの視界から外れたモデルにはベベル（斜面）がある。これは、そのような「装飾」を処理する際に、追加のハードウェアリソースが割り当てられることを意味するが、プレイヤーはそれに気づくことはない
- ディテール不足。このタイプの欠陥は、過剰なディテールの欠陥の正反対である。もしモデルがプレイヤーにはっきりと見えるのであれば、角ばった印象を与えない程度にディテールがあるべきである
- レンダリングされないジオメトリの部分。例えば、モデルから必要なパーツが欠落していると、非現実的なものになる

モデルのテクスチャリング

この段階で、テクスチャーをアーティストまたはアートディレクターが受け入れる。この場合、アプリケーションのテクスチャーの正しい適用、継ぎ目や引きつれがないこと（つまり、ポリゴンのトポロジーが正しくないこと）は発見されないかもしれない。このテストは、一般的なカラーパレットと汚れのレベル（汚れの存在、不要な色の挿入、モデル上の異なる色のランダムピクセルなど）のみに関係する。この段階で発見される典型的な欠陥は、自動マッピングとそれに続くテクスチャリングが共通のエッジを持つモデルの部分に異なる色のテクスチャーが適用されるような方法で実行され、その結果継ぎ目が目立つ場合である。

オブジェクトの LoD とコリジョンモデルのレビュー

この段階では、アーティストとレベルデザイナーの両方がモデルの品質評価に関与する。レベルデザイナーは、ゲームプレイへの影響という観点からコリジョンモデルを評価する。例えば、特定の場所で詳細なモデルを使用する必要があるのか、それとも簡略化できるのかなどである。この場合の典型的な欠陥は、詳細すぎる（例えば、コリジョンモデルの許容ポリゴン数を超えている）、または十分に詳細でない（例えば、十分に詳細でないコリジョンモデルを含んでいる）モデルである。

ゲームエンジンへのモデルのエクスポート

これは、LoD を切り替える距離や、オブジェクトのヘルスポイント数の設定が正しいかどうかを検討するものである。テストは、テスト担当者がビデオゲームエンジンのコンテンツエディターとゲームアプリケーションの両方で直接行う。同時に、アーティスティックテストのほとんどは、テスト対象の視覚的な評価に還元され、多くのテストは、アーティストやモデラーのテストと重複する。テストでは、以下のことが行われる：

- オブジェクトが使用される場所や状況を考慮した、オブジェクトの LoD の切り替えの可視性
- プレイヤーにとって目立つオブジェクトのジオメトリに隙間や継ぎ目
- さまざまなグラフィックオブジェクト上の碑文の可視性
- モデルの破壊効果（例えば、色やサイズを考慮する）
- オブジェクト・アニメーションにおける視覚的欠陥

アーティスティックテストでは、プレイヤーは直接参加することができ、コンテンツを視覚的に評価し、その他のグラフィックの欠陥を発見することができる。

グラフィックのテストは、ゲームマップのテストと一緒に行われることが多い。この場合、さらにいくつかの段階がある。

地図上のオブジェクトの配置

これはテストにとって最も重要な段階の 1 つである。オブジェクトの配置はレベルデザイナーによって行われる。このプロセスで、典型的な欠陥として、オブジェクトがマップ（地形）の下に「ぶら下がったり」、凹んだりすることがある。これは、地形のジオメトリを変更したり、オブジェクトを露出させ

たり、マップ自体を編集したりする可能性のある多くの専門家が並行してマップを扱う場合によく起こる。

エフェクトの作成と配置

この段階では、エフェクトアーティストが新しい特殊効果の作成に取り組む。さまざまな状況でそれを見ることで、芸術的、技術的なレンダリングの欠陥を検出することができる。このようなテストの例として、エンジン内のエフェクトの視覚的テストがある。エフェクトアーティストは、ゲームクライアントの各エフェクトを見て、誤ったエフェクトパラメータ（エフェクトの減衰時間が不十分など）を特定する。

グラフィックのテストは、しばしばプレイテストで行われる。これは、マップをゲームの状況でテストするすべてのテストに関連する。このテストの目的は広範であり、制作のどの段階にも適用される（例えば、マップでのゲームプレイの評価を行う）。制作の最終段階に近づくと、プレイヤーの視点から見たマップの芸術的な評価やゲームプレイの評価が含まれることもある。ゲームデザイナーがマップのバランスを調整するために必要な統計情報の収集に加えて、オブジェクトのぶら下がりからグラフィックのレンダリングの欠陥（フリーズ、不具合、故障など）に至るまで、さまざまな芸術的、技術的な問題が明らかになる。純粋にアーティスティックテストでは、プレイテストによってプレイヤーの視点から顕著な問題を明らかにする。

地図のテスト

これはテストの最終段階であり、マップは完全に準備され、これ以上の作業は行われない（稀な例外を除く）。テスト担当者は、マップ妥当性確認チェックリストに基づき、ぶら下がっているオブジェクトの確認からアニメーションのテストまで、完全なマップ妥当性確認を行う。テスト担当者はこのようなテストに集中することで、アーティストに比べてより多くの欠陥（例えば「ぶら下がった」オブジェクト）を検出することができる。アーティストのテストでは、アーティストはマップを視覚的にインスペクションするが、テスト担当者は、グラフィックオブジェクトのより問題のある課題箇所を検出するために、特別なツールセットを使用する。

3.3.2　グラフィックの歴史的正確性のテスト

グラフィックコンテンツのテストとは別に、歴史的な正確さのテストがある。歴史的な正確さは、歴史的正確性と歴史的信憑性の両方を考慮する。

歴史的信憑性は、歴史上の特定のエピソードが特徴的な画像を用いて描写されるゲームの品質に影響を与える。これらは特定の出来事や人物などには言及しない。

歴史的正確性は、歴史上の特定の時代を描写するゲームの品質に影響する。すべての重要な出来事は、その時代の特徴的なイメージとともに、関連するすべての人物とその行動を含めて、そのダイナミクスに描写される。その時代とは異なるイメージは基本的に排除される。

ゲームでは原則的に、夢中にさせるゲームプレイに重点が置かれるため、歴史的信憑性を維持することは不可能である。最初から結果が決まっているゲームにプレイヤーを巻き込むのは難しい。

しかし、歴史的正確性は、特にゲームのプロットが特定の歴史的時代と結びついている場合、特別な注意を必要とするゲームの分野である。

グラフィックコンテンツの歴史的正確性をテストする際に考慮すべき典型的な点は以下のとおりである：

- 登場人物が歴史的原型に似ていること
- 建築物を再現する際の正確さ
- 特定の時代の武器、装備、乗り物、衣服とその特徴を正確に再現すること
- 日常生活の再現精度
- 歴史上の出来事や日付の正確さ

歴史的な正確さテストを行う場合、テスト担当者は幅広い知識と視野を持つべきである。可能な限りリアルなゲームを作るために、開発者が歴史コンサルタントをテストに参加させることはよくある。

3.4 グラフィックステストのサポートツール

コンテンツをテストする際、グラフィックスのテスト担当者は、ゲーム内のコンテンツエディターから自動化ツールやテストスクリプトまで、さまざまなツールを使用する。原則として、ゲームは、オブジェクトエディター、ワールドエディター、ライティングエディターなど、グラフィックステスト用のツールがいくつか組み込まれたビデオゲームエンジンを使用して作成される。

すべてのツールは、プレイヤーが既存のコンテンツとインタラクションできるようにカスタマイズする機能を提供する。モデルエディターは、エンジンとのインタラクション時に発生するさまざまなトリガーやサウンド、その他のイベントなど、モデルに効果的なバインディングを可能にする。ワールドエディターは、マップの作成、オブジェクトの配置、マップのゲームプレイメカニクスのカスタマイズを可能にする。テスト担当者がほとんどのテストを行うのは、このエディターである。その際、開発者と同じツールを使用する。これらのエディターは、コンテンツのセットアップとテストを可能にする広範なツールキットも提供する。

専用ツールは通常、グラフィックカードメーカーが開発する。これらのツールは、2D および 3D グラフィックスを作成するために使用されるさまざまな API セットを使用して、ゲーム内のフレームのキャプチャーや任意のアプリケーションの詳細な分析を可能にする。また、以下のような評価も可能である：

- ゲームフレームがどのように準備されているか（例：ジオメトリ、テクスチャー、ドローコールなど）
- パフォーマンスの問題が発生した場合
- 実行中のジオメトリー
- 3D モデラー、グラフィックアーティスト、アニメーターのためのデバッグ

これらの各ツールの使用法の詳細は、特定のソフトウェア仕様に記載されている。

グラフィックステストツールには、ゲーム内の所定のルートに沿ってキャラクターを自動的に通過させる自動化スクリプトが含まれる。特に性能テストや互換性テストに使用される。

4. サウンドテスト - 190 分 (K3)

ビデオゲームに特化したキーワード

アンビエント、バイノーラル効果、ディストーション、ラグノイズ（フォーリー）、オクルージョン、リバーブ、スキッピング、音の不連続性、効果音、サウンドゾーン、ボリューム

第 4 章の学習目標

4.1 ゲームプロダクトのサウンドコンテンツの特色
GaMe-4.1.1 (K1) ゲームプロダクトのサウンドコンテンツの特色を想起する。

4.2 サウンドコンテンツの欠陥の種類
GaMe-4.2.1 (K1) サウンドコンテンツの欠陥の種類を想起する。
GaMe-4.2.2 (K2) サウンドコンテンツの欠陥を分類する。

4.3 ゲームプロダクトにおけるサウンドコンテンツのテストへのアプローチ
GaMe-4.3.1 (K2) オーディオコンテンツのテストへの主なアプローチを要約する。
GaMe-4.3.2 (K2) 音楽とサウンドのミックスをテストするための主なアプローチを要約する。
GaMe-4.3.3 (K2) 作曲テストの主なアプローチを要約する。

4.4 サウンドテストの実行
GaMe-4.4.1 (K2) 音楽コンテンツのテストレベルについて説明する。
GaMe-4.4.2 (K1) サウンドをクライアントに統合することの特色を想起する。
GaMe-4.4.3 (K1) サウンドテストの責任範囲を想起する。
GaMe-4.4.4 (K3) サウンドテストへのアプローチを適用する。

4.5 サウンドテストのサポートツール
GaMe-4.5.1 (K2) サウンドテストツールの使用法を要約する。

4.1 ゲームプロダクトのサウンドコンテンツの特色

現代のビデオゲームにおいて、サウンドは重要な要素である。サウンドや音楽は、適切なムードを作り出し、危険を知らせ、キャラクターの感情を伝え、ゲームの世界の全体像を補完し、作り出す。

まったくサウンドのない映画やビデオゲームに直面した人は、わずかな違和感を覚える可能性が高い。これは、脳が聴覚を活性化させようとしているにもかかわらず、それができず、代わりに目の前で起こっていることについての知覚エラーを知らせるために起こる。

低品質、非現実的、不適切、または機能性の低いサウンドは、プレイヤーをイライラさせ、ゲームプレイへの没入感を壊す可能性がある。

ビデオゲームのサウンドトラックを制作するプロセスは、ビデオゲームのコンテンツ構成の複雑度を考慮する必要があるため、非常に長く時間がかかる。

- 環境音
- キャラクターの声
- バックグラウンド・サウンドトラック
- 効果音
- さまざまなサウンドオブジェクト
- 音声ファイル

これはテスト作業を複雑にしている。

4.1.1 サウンドの種類

ゲームのサウンドデザインは、ビデオゲームによってさまざまであり、その目的もさまざまである。どのような種類の音を区別し、どのような機能を果たし、どのような目的で作られるのが通例なのかを検討していく。

音楽

ゲームの名刺代わりとなる主要な音楽テーマは、最もシンプルなゲームにも登場する。古い 8 ビットのビデオゲームの曲は、長い間、古典となっている。

音楽の主な機能

- その瞬間の「エピックさ」を強調する
- イベントのダイナミクスを高める
- 雰囲気を高める
- プレイヤーを適切な気分にさせ、キャラクターの感情の状態を説明する
- 繰り返されるメロディーによって、無意識のうちにプレイヤーの中にある種の感情的な反応が形成される

効果音

ゲームの世界に主人公が関与できるオブジェクトがたくさんある場合、そのひとつひとつにリアルなサウンドが必要である。それによって、ゲームの世界が生き生きとしたものになり、真実味が増す。

ドアの開閉、打撃、射撃と武器のリロード、救急箱の使用、燃料保管容器の爆発、ガラスの割れる音など、これらすべてが適切なタイミングで音声化されて再生される。

小規模なゲームでも、開発者はインターフェース要素（ボタン、スイッチ、メニュー項目）と関連する効果音を追加する。これはプレイヤーにフィードバックと、プレイヤーの行動が知覚および解析されていることの理解を与える。

キャラクター・サウンド（フォーリー）

ゲーム中のキャラクターが発する音には、衣擦れ音、呼吸音、叫び声、足音、うめき声などがある。この用語は、効果音のパイオニアの一人であるジャック・フォーリーにちなんで名付けられた。シューティングゲームなどの一部のゲームでは、このような音はキャラクターが負傷していることを示す指標になりうる。これは、ポップアップで表示されるダメージの数字よりも、プレイヤーにとってわかりやすい。

音声

キャラクターによって作り出される音声は、個別で強調しておいたほうがよい。現在のゲームでは、ノンプレイヤーキャラクター（NPC）ごとに、プロの俳優によって録音された固有の音声が用意されている。質の高いセリフを収録することで、キャラクターの感情を正しく伝え、キャラクターとプレイヤーの間に感情的な接続性を生み出すことができるため、開発者は吹き替えに多くの資金と労力を投入する。

アンビエント・サウンド

アンビエントサウンドとは、特定の場所のサウンドの特徴を表すものである。音楽テーマと同様、アンビエントは雰囲気を強調し、プレイヤーに必要なムードを作り出す。アンビエントは、そのサウンドがどこから来るのかを推測させるものではない。特定の場所、状況、またはゲームサイクルのステージに単に付随する環境音である。

アンビエントサウンドは通常、バックグラウンドで再生され、キャラクターのアクションに依存しない。

アンビエントサウンドの例

- 森の葉の擦れや鳥のさえずり
- 海の波しぶき
- 酒場の客たちの不明瞭な会話
- パトカーのサイレンや通過する車の騒音など

これらはすべて、プレイヤーの空間内の移動や、ゲームの周囲の世界をより認識することに役立つ。

4.1.2 効果音とテクノロジー

サウンドトラックは、現代のオーディオ機器やテクノロジーの性能を考慮し、信憑性のあるエフェクトを生み出すためにサウンドエディターで処理されることが多い。

ゲームの仮想世界にユーザーを没入させるためには、リアルなサウンドだけでは不十分な場合がある。発生していることの現実感を高めるために、開発者は一部のサウンドを誇張することがよくある。このようなテクニックは、例えば映画の格闘シーンでよく見られ、打撃や格闘家の動きのサウンドまでもが、現実よりもはるかに大きく聞こえる。

このエフェクトを実現するために、さまざまな技術やテクノロジーが用いられている。

オクルージョン

最も一般的なテクニックの 1 つは、サウンドが障壁（壁など）を通過するときのオクルージョン効果である。このような状況で音の専門家は、よりリアルにするために、単に音量を下げるだけでなく、特別な効果を加える。

音源とリスナーの間に何もない壁がある、別の部屋にあるが物体が直接見える（出入り口）、同じ部屋にあるが物体が見えない（物体の間に柱がある）といった場合には、主音と反射音は異なって認識される。

反響音

反響音効果は、音量や空間の奥行きを（反響から）伝えるために使われる。

この場合、密閉された空間で音源から発せられた音が壁で反射して無数のエコーを起こし、徐々にフェードアウトしていく。キャラクターが洞窟や狭い部屋、音が反響しやすい部屋にいる場合に効果的である。

高度なサウンドエンジンを使えば、銃声、爆発音、声、足音などにエコーをかけたり、音量を上げたりすることが簡単にできる。

バイノーラル効果

バイノーラル効果（ラテン語の bini（2 つ、対）と auris（耳）に由来する）とは、人が 2 つの耳で同時に聴き、首を横に向けると、音が片方の耳に先に届くという事実に基づいている。このおかげで、人は音源がどの方向にあるのかと、その距離さえも判断できる。

この効果を伴った音を録音するには特殊なマイクが使われ、効果を認知するにはヘッドホンの使用が推奨される。これは、音が特定の方向からそれぞれの耳に入ってくるようにするために必要なことである。サウンド環境は没入感をより深くし、プレイヤーを助けてくれるかもしれない。バイノーラルオーディオのおかげで、プレイヤーは音がどちらから聞こえてくるかわかる（例えば、敵がその瞬間にどこに現れるか）。

4.1.3 サウンドエリア

プロジェクトによっては、サウンド環境をさまざまな場所に分散させるために、いわゆるサウンドエリアが使われる。

ゲームでは、ロールプレイングゲームのキャラクターが村に入ると特徴的な音が現れたり、ストラテジーゲームでプレイヤーが基地にカメラを近づけると、資源を集める音が聞こえたりする。

サウンドゾーンには移行が必要である。移行の際には、一方のゾーンのサウンドがわずかにフェードし、もう一方のゾーンのサウンドが現れるはずである。

実際の生活におけるサウンドゾーニングの有効性は、部屋の窓を開けたり閉めたりすることで観察できる。窓を開けると、通りの音ははっきりと聞こえ、窓を閉めると、まったく聞こえないか、または極端に小さくなる。

4.2 サウンドコンテンツの欠陥の種類

サウンドコンテンツの欠陥には 3 つのカテゴリーがある

- サウンドがない
- 間違ったサウンド
- 不正なサウンド再生

効果音の消失

キャラクターの特定のアクションや特定の瞬間に、あるべきサウンドが再生されない。

間違ったサウンドの再生

例えば、大砲の発射音がピストルの銃声のように聞こえたり、モーターボートが飛行機のように聞こえたり、キャラクター（ゲーム内外を問わず）が他人のセリフを言ったりする。

例えば、ゲームキャラクターが現代的な語彙を使ったり、映画や音楽の引用をしたり、T-34 戦車が T-72 戦車のように聞こえたりするなど、効果音が歴史的信憑性を侵害する。稀なケースだが、テスト対象のゲームの重要な価値としてリアリズムと信憑性が明記されている場合は、意味がある場合がある。

正しいサウンドの不正な再生

オーディオ・ドロップ

このような欠陥が発生すると、接続の悪い電話をしているときのように、再生されているサウンドの一部が消えてしまうことがある。このため、プレイヤーは、例えばダイアログのフレーズの意味を誤解する可能性がある。再生されているのがこの音だけであれば、テスト担当者は簡単に問題を発見することができる。しかし、複数の音が同時に流れていて、そのうちの 1 つが消えてしまった場合、そのような欠陥を検出するのはずっと難しくなる。

スキッピング

音楽やオーディオエフェクトのスキッピングは、サウンドファイル自体の損傷だけで発生するものではない。スキッピングは多くの場合、パフォーマンスの問題によって引き起こされる。この場合、フレームレートが失敗すると発生する。

ディストーション

パフォーマンスの問題により、一部のフレーズが歪んで聴こえない場合がある。

再生ラグ

ゲームでは、サウンドデザインがアニメーションやオブジェクト、ゲームの状況とのラグが発生する瞬間がある。

サウンドコンテンツの欠陥の例と考えられる原因

1.オブジェクト/環境のサウンドが欠落

欠陥：金属面を移動する際、足音がせず、キャラクターが静かに移動する。

原因：開発者がこの金属面へのサウンドの設定または関連付けを忘れた。

2.オブジェクトや周囲のサウンドが大きすぎる／小さすぎる

欠陥：ある場所の特定のオブジェクトが燃えているのだが、火の音が大きすぎたり、逆に非常に小さかったりして、非現実的に見える。

原因：エフェクトのボリュームが正しく設定されていない。

3.オブジェクトや環境のサウンドが正しく設定されていない

欠陥：水上でモーターボートを運転中、レーシングカーの音が流れる。

原因：オブジェクトのサウンドファイルが正しくない。

4.ソースによっては音の位置が正しくない。

欠陥：ゲームにはハンマーで刀を打ち続ける鍛冶屋が登場するが、その打撃音は部屋の別のコーナーで再生される。

原因：サウンドエディターで、効果音が間違った位置に設定されている。

5.サウンドカバレッジエリアが正しく設定されていない

欠陥：銃声が半径数メートルの範囲でしか聞こえない。プレイヤーがそれ以上離れると音がまったく聞こえなくなり、非現実的である。

原因：音の届く範囲が正しく設定されていない。

6.サウンドのディストーション

欠陥：音割れやヒスノイズが発生する。

原因：破損したオーディオファイルが使用されている。

7.サウンドの繰り返し。

欠陥：サウンドがループして連続再生される。

原因：サウンドループを停止するポインターが正しく設定されていない。

8.とぎれとぎれで再生されるディストーション

欠陥：ゲームキャラクターの移動中に、移動のサウンドでディストーションが発生する。

原因：サウンドアセンブリーの問題、またはゲームクライアントが全体的に不安定なことが原因で問題が発生する可能性がある。

9.オーディオ再生のタイムディレイ

欠陥：カットシーン中、キャラクターが話し始めるが、そのセリフは唇が動き始めるより少し遅れて再生される。

原因：キャラクターの唇が動くアニメーションが、対応するサウンドファイルの開始位置とずれている。

4.3 ゲームプロダクトにおけるサウンドコンテンツのテストアプローチ

ゲームのサウンドデザインは、効果音、キャラクターの声、または音楽の形で表現される。ゲームのオーディオコンテンツのテストは、フォーマルな方法とインフォーマルな方法で実施される。フォーマルなサウンドテストのアプローチには、すべてのオーディオファイルのプレゼンス、正しさ、ボリューム、完全性のテストが含まれる。インフォーマルなサウンドテストは、ゲームのリアルさや雰囲気の維持を評価するものである。例えば、ホラージャンルのゲームでは、状況を強調し、プレイヤーを定期的に緊張状態に維持するために、その場所に「重い」音楽を伴奏させる。

テスト担当者は、事前にレビューしたサウンドオブジェクトの仕様書からテストケースを導出する。この仕様から、テスト担当者はサウンドがどのような場所、ゲームシーン、シチュエーションで再生されるべきかを明確に理解する必要がある。場合によっては、オーディオデザイナーと相談し、作成されたサウンドコンテンツの具体的な詳細を知ることが必要となることもある。また、テスト担当者がビデオゲームリソースエディターを使って、ゲーム内のプレイヤーのアクションや環境とサウンドファイルの同期をテストする場合もある。

テスト担当者によるゲームのサウンド設計のテストには、いくつかのステップがある。

4.3.1 オーディオコンテンツのテスト

テスト担当者はサウンドを聞き、サウンドエディターまたはビデオゲームリソースエディターを使って、作成したコンテンツのオーディオパラメータをテストする。すべてのサウンドファイルをテストする必要がある。

これは最も広範囲に及ぶ段階の 1 つである。以下の音の特性が要件と照らし合わされる：

- 対象物や環境に対する適切さ
  例えば、キャラクターが鉄の鎧を着用している場合、彼が動くと金属の音が布の音よりも優先されるはずである。ダイナマイトの爆発音や落ちた岩の音がかすかに聞こえ、砂の上を移動する時に歩道を踏みつける音のほうが明らかに聞こえたりすると、バーチャル世界への没入感が損なわれる
- 俳優の声
  すべての音声フレーズにおける俳優の声は、それを話すキャラクターとゲームの設定に対応していなければならない。通常、声は俳優の性別や年齢に合わせなければならない。これは全体的な雰囲気を維持するために行われるもので、例えば南部出身のキャラクターは現代のアイルランド語ではなく南部訛りがある
- 不快ではないこと
  複数の音が一緒になったサンプルは、ときとして人間の耳に不快感を与えることがある
- ボリュームレベル
  1 つのゲーム内で複数のファイルを使用する場合、各サウンドファイルのボリュームは同じでなければならない
- 効果音

  サウンドとナレーションの両方に適用される効果音は、正しく使用されなければならない。場所やシーンがさまざまなディテールで満たされている場合、ナレーションによってディテールが強調される。大きなオフィスでは、システムの動作音、プリンターのブーンという音、エアコンの音、キーボードを押す音、従業員の会話などが聞こえる

すべてのオブジェクトに丁寧な録音が施され、リアルな場面作りに貢献している。例えば、プレイヤーが村にいると自覚したとき、牛の鳴き声が聞こえるとは限らないが、目の前に製材所が現れたら、製材所の特徴的な音を期待するだろう。森の中の小さな小川にはサウンドデザインがないかもしれないが、主人公が滝の近くに立った場合、水の音が聞こえないのは欠陥がある（または設定でサウンドがオフになっている）としか説明できない。

4.3.2 音楽とゲームサウンドのミックスのテスト

プロジェクトの規模や開発チームの専門スキルはプロジェクトによって異なるため、役割によって果たす機能を明確に分けることはできない。声優が必要でない場合もあれば、フォーリーアーティスト（ノイズを作る専門家）の仕事をオーディオデザイナーに任せる場合もある。あるいは専門家自身が既成のサンプルを作成または購入し、それをクライアントに実装し、サウンドのテストをする場合もある。

オーディオデザイナーの役割は、サウンドを作成し、オーディオファイルを録音することである。これらのオーディオファイルは、さまざまな用途でテストする必要がある。アーケードゲームのサウンドシステムと携帯電話のサウンドシステムは大きく異なるため、オーディオのテストが必要である。コンソールゲームは、小型のテレビ内蔵スピーカーとホームシアターシステムの両方で正しく再生されなければならない。

高品質で一見適切なサウンドであっても、最終的なミキシングが正しく行われなければ、プレイヤーに混乱や苛立ちを与える可能性がある。例えば、アクションの音がアクションそのものよりもずっと長く続いたり、ボリュームが適切でないなどである。

テスト担当者は、ビデオゲームエンジンに組み込まれたリソースエディターを使用するか、自分の経験または提供されたドキュメントに基づいて、完成したサウンドピクチャと、クライアントに新しいサウンドの変更を加えることの正しさをテストする。

ゲームプレイ中、プレイヤーはさまざまなオブジェクトから発せられる多くの音や、バックグラウンドで流れる音楽を耳にするため、すべてのオーディオコンポーネントが互いに調和していることが重要である。テスト担当者は、プロジェクトのサウンドコンポーネントを評価し、音質や自然さと合わせて、可能性のある逸脱や音量に関する非現実的な点を常識に基づいて特定する必要がある。また、音源とオブジェクトの位置関係の正確さを評価することも重要である。例えば、ゲームの中でキャラクターがドアを開けるとき、ドアが開く音をドアからではなく、キャラクターの背後から間違って聞こえてしまうなどである。また、テスト担当者が、画面上で起こっていることとサウンドのタイミングを正しく評価できることも重要である。

4.3.3 ミュージックコンポジションのテスト

コンポジションまたはミキシングは、個別に録音されたトラックから最終的なレコーディングを作成する段階である。これはオーディオ録音の次のステップで、元の録音トラックを選択、編集（ときには復元）し、1 つのプロジェクトに組み合わせ、効果音との処理を行う。編集は多くの場合、単独の作業段階である。

電子音楽を使用するゲームにおけるミキシングは、オーディオコンテンツ制作の次のステップである。電子音楽のプロジェクトでは、サウンドレコーディングの段階がないことがほとんどである。電子音楽の制作とミキシングの境界線は曖昧である。多くのバーチャルシンセサイザーには、さまざまなサウンドのテンプレート処理がすでにあるため、プロジェクトはすでに部分的にミキシングされている。

その結果、マルチチャンネルのプロジェクトはモノラル、ステレオ、マルチチャンネルのフォノグラムに出力され、通常マスタリングと呼ばれる工程で最終的な形になる。

マスタリングが完了すると、マスタリングされた音楽のテストが始まる。テスト担当者は、音楽コンポーネントに欠陥がないか、音の不連続性や干渉をテストし、故障が発見された場合は報告する。

4.4 サウンドテストの実行

4.4.1 ゲームソフトウェア開発ライフサイクルにおけるオーディオ音楽コンテンツのテストの段階

多くの場合、サウンドテストは開発の後半に始まる。開発者がゲームのコンテンツ（オブジェクトやキャラクターのモデル、マップ、アイテムなど）を作成する一方で、サウンドコンポーネントは 後回しになる。開発段階ではオブジェクトにサウンドがなくても、オブジェクト自体にカスタマイズされたアニメーションを含む完成モデルがあり、それがクライアントに統合され、テストできる状態になっている。

サウンドサンプルは、サウンドを制作したスペシャリスト（オーディオ設計者、フォーリーアーティスト、効果音のスペシャリストなど）が事前にテストする必要がある。彼らは個々のサウンドのテストに参加し、さらにクライアントに統合するために開発に送ることもある。

4.4.2 ゲームへのサウンドの統合

サウンドデザインをゲームに統合した後、テスト担当者はサウンドの以下の特徴を評価する：

- サウンドおよび/または音楽のオン/オフ機能
- 音量変更機能
- オーディオファイル名の単一フォーマットと、プロジェクト内のフォルダへの配置ルールへの適合
- ゲームマップ上の音源の位置、音が伝播する範囲
- 統合されたサウンドによるシステムパフォーマンス
- ヘッドホン、スピーカー、モニタースピーカーなど、異なるオーディオシステムでの音の互換性

4.4.3 関係者の責任範囲

プロジェクトにおける音楽とサウンドデザインの制作プロセスには、特定の機能を持った、いくつかの異なるタイプの専門家が関わっている：

役割	責任
作曲家	ゲームのサウンドや音楽の制作に携わる。
オーディオ・エンジニア	録音、音響増幅、放送機器の運用に携わる。
サウンド・エンジニア	サウンドの専門知識を最大限に活かし、各オブジェクト、ゲームシーン、ゲーム全体の最終的なサウンドピクチャを決定するスペシャリスト。
オーディオ・デザイナー	録音された音（サンプル）でオリジナルのオーディオファイルからコンテンツを作成し、サウンドエディターで調整する。
フォーリーアーティスト	現実に存在する音をより鮮明で豊かなものにし、現実には存在しない効果音を生み出す。通常のオブジェクトの音を組み合わせてサンプルを録音し、そこから例えばレーザーソードがうなる音、ゾンビがうなる音、扉がバタンと開く音などの有効性を作り出す。

役割	責任
声優	聞こえるレプリカを持つゲーム内のキャラクターに命を吹き込むために、自分の声で演じる。
開発者	作成およびカスタマイズされたオーディオコンテンツをゲームクライアントコードに統合する。
レベルデザイナー	音源をレベルごとに分配し、音源にサウンドファイルを関連付ける。
物語デザイナー	コンピューターゲームの開発において、ストーリーや物語を担当するスペシャリスト。ストーリー展開に関わる場面（カットシーン、イントロ、ダイアログ、ゲーム中の重要な場面など）で流れるサウンドや音楽を担当する。
ゲームデザイナー	ゲームドキュメント（ビデオゲームの設計書など）の作成に携わる。このドキュメントには、ゲームのルールや特徴が簡潔な言葉で記述されている。こうして、エンジンが開発される前から、ゲームデザイナーはゲームの全体像を作り上げていく。開発過程では、「コンサルタント」としての役割もあり、プロジェクトの主要な考え方への提案や変更の適合性を承認する。また、ゲームのテストにも個人的に参加する。
テスト担当者	完成したサウンドピクチャと、クライアントでの新しいサウンドの変更の正しさをテストする。

4.4.4 サウンドオブジェクトのテストにおける手順とアプローチ

サウンドオブジェクトをテストする場合、テスト担当者は特定のバージョンのゲームでサウンドがどの程度適切に設定されているかの完全な情報を把握するために必要な手順やアクションを実現する必要がある。アクションのリストは、テストが必要なオブジェクトに直接依存する。例えば、クライアントに追加された武器のサウンドデザインの最終テストでは、テスターは多くのアクションを実行する：クライアントで初期状態では利用できない場合に備えて武器の追加、テストマップへの新しいコンテンツのロード、武器に付随するすべてのサウンドを最大限にテストするために必要なアクションの全リスト（発砲、リロード、再リロード、発砲モードの切り替えなど）など。

各オブジェクトに対する手順やアプローチのリストは異なることを理解することが重要である。例えば、ある環境でテストされているオブジェクトがキャラクターに影響するオブジェクトでない場合、テスト担当者はサウンドデザインそのもの、例えばキャラクターが近づいたときのせせらぎや川の音をテストする。テスト担当者は、特定のオブジェクトで利用可能なすべてのサウンドに関する情報を理解しておく必要がある。また、サウンドコンテンツの統合をテストし、キャラクターやゲーム環境のアクションと同期させるために、ビルトインのリソースエディターやサウンドエディターを活用する必要がある。

4.5 サウンドテストのサポートツール

ビデオゲームエンジン・オーディオエディター

テスト担当者は、常にビデオゲームエンジンのオーディオエディターにアクセスできるとは限らない。そのため、テスト中は、コンテンツの妥当性確認のための標準的なソフトウェアセットを使用する。このようなエディターは、開発チーム内で働くテスト担当者（いる場合）、またはオーディオエディター

で直接作業する専門家（作曲家、オーディオデザイナーなど）が使用する。オーディオデザイナーはテストに参加しなければならない。彼らは、レベルのバランス、イコライザーの調整、すべてのアセットのミキシング/アライメントなどを一緒または単独でアセットのテストを行っている。アセットがゲームの作業バージョンに追加されるとすぐに、オーディオデザイナーはそれらを簡単に入手し、必要に応じて編集できるようになる。

サウンドコンテンツのメンテナンスを容易にするため、すべてのサウンドサンプルは異なるサウンドトラックに収め、フォルダに整理し、名前を付け、タイムスタンプとバージョン管理を行わなければならない。また、サウンドアダプターのハードウェアリソースに負担をかけないように、オーディオトラックを圧縮されたデジタルフォーマットに変換することも効果的である。

マップ／ロケーションエディター

テスト担当者が開発ソフトウェア（多くの場合、マップ/ロケーションエディター）を利用する場合もある。これにより、テスト担当者はオーディオコンテンツやクライアント内のすべてのコンテンツをテストする際に、その機能を拡張することができる。ゲーム開発者は、レベルマップ上で直接「シミュレーションされた」ゲームプレイをサポートする機能をマップエディターに追加する。そのため、テスト担当者は、マップエディターからゲームクライアント自体に切り替えることなく、必要なテストを実施することができ、時間を大幅に節約することができる。

マップ上で利用可能な効果音と音源のほとんどは、レベルエディタを使って配置・調整する。マップエディターでは、テスト担当者がクライアントロケーション上の音源の位置を確認でき、ロケーション内のさまざまなオブジェクトやキャラクターと関連付けることができる。そして/または、専門家がエディターを使って音源をテストする際、再生されるサウンドとサウンドの伝搬距離を、球体として、またはテスト担当者やエンドユーザーがその境界を越えたときに、紐付けられた音源から再生されるサウンド/音楽が聞こえ始める "ゾーン" として表示し、テストすることができる。

専門家がエディターを使って音源をテストする際、サウンドとサウンドの伝搬距離をテストすることができる。これはテスターやエンドユーザーがその境界を越えたときに紐付けられた音源から再生されるサウンド/音楽が聞こえ始める球体または ”ゾーン” として表示されている。

サウンドの伝搬が "ゾーン" の形で実装されている場合、テスト担当者は、 "ゾーン" の境界を越えたときに音源から再生されるサウンドをテストする。また、音源に近づく際の音量の上昇をテストすることも現実性を持たせるために重要である。球体の音の伝搬距離テストは、伝搬ゾーンの形だけが異なる。また、このエディターでは、各クライアントの音源の結合、位置、それぞれの構成をテストすることができる。

この情報を使って、テスト担当者はサウンドコンテンツのリアルさ、サウンドの伝搬距離、サウンドの正しさ（場所、オブジェクト、キャラクター）を評価し、欠陥が検出された場合には開発者に指摘することができる。例えば、特定の音源の距離が正しく設定されていないという欠陥の説明を作成する。または、特定の音（特定の音源を指す）のボリュームの音量が、 "隣接する" 音源の背景に対して大きすぎる。欠陥の記述において、テスト担当者はエディターを通して作業することで、開発者の作業を容易にし、特定の問題を指摘することで問題箇所を発見しやすくすることができる。

5. ゲームレベルテスト - 65 分 (K2)

テストキーワード
プレイテスト

ビデオゲームに特化したキーワード
カラーコーディング、ビデオゲームレベル、侵入不可能領域、レベルエディタ、レベルプロトタイプ、物語、構造的ジオメトリ、トリガー

第 5 章の学習目標

5.1 ゲームレベルデザインの原則とコンセプト
GaMe-5.1.1 (K1) ゲームレベルのコンポーネントを想起する。
GaMe-5.1.2 (K2) ゲームレベルの典型的な欠陥を分類する。

5.2 ゲームレベルテストの段階と実行
GaMe-5.2.1 (K2) ゲームレベル作成のさまざまな段階で実施されたテストを要約する。
GaMe-5.2.2 (K2) ゲームレベルのテストに参加する専門家の責任範囲を比較する。

5.3 ゲームレベルのテストをサポートするツール
GaMe-5.3.1 (K2) ゲームレベルをテストするためのツールの使用法を要約する。

5.1 レベルデザインの原則とコンセプト

5.1.1 ゲームプロジェクトのジャンルによる「レベル」という用語とその特殊性

ゲームレベルとは、ゲームの仮想世界の独立した領域のことで、プレイヤーはそこで、宝を見つける、すべての敵を倒す、または単に出口に到達するなど、特定のタスクを完了する必要がある。レベルを完了するために必要な一般的なタスクは、場所と表裏一体であり、多くの場合、レベルを完了するために必要な多くの小さなタスクやクエスト、またはサイドクエストで構成されている。多くのビデオゲームエンジンでは、レベルはロード画面によって分割される。

レベルデザインは、マップ、場所、タスク、ミッション、およびゲームのレベルのその他の環境の作成に関連するゲーム開発の段階である。レベルデザインには、ゲームオブジェクトやゲームメカニクスの外観、プレイヤーのパス上にある障害物、ゲームのストーリーライン、および意図されたゲーム体験を総体的に作成するその他の要素が含まれる。

レベルを作成するには、通常、特別なソフトウェアが使用される - レベルエディタ。

ゲームレベルは、1 つの全体の中で互いに接続された多くのコンポーネントから構成される。主な構成要素は以下のとおりである：

構造的ジオメトリ

構造的ジオメトリまたは基本レベルのジオメトリは、あらゆるゲーム空間の最も重要で形成的な要素の 1 つである。構造的ジオメトリは、地形の起伏とゲームキャラクターが移動できる表面を設定する。

構造的ジオメトリも、キャラクターが移動できるレベル空間を制限する。例えば、レベルの境界に通れない山やその他の地形がある場合がある。

レベルに明確にマークされた入口と出口がある場合、構造的ジオメトリは、プレイヤーがレベルの最後まで行くべき場所のガイダンスを提供することができる。

ゲームの舞台となる場所によって、さまざまな要素がレベルを形成するオブジェクトとして機能する。例えば、街の中であれば、建物、道路、橋、地下道などであり、街の外であれば、野原、丘、岩、渓谷、洞窟などの自然景観要素である。

ゲーム環境

ゲーム環境オブジェクトは、レベルデザイナーによって必要とされるゲーム空間をより信憑性の高いイメージにするために使用される。

例えば、村の通りは、家、建物、道路で構成されているため、デザイナーはそこにさまざまな大きさのオブジェクトを配置する。大きなものでは、木、塀、荷車、中くらいのものでは、井戸、樽、ベンチ、小さなものでは、庭の植物、石など 。

照明

さまざまなタイプの光源を使用することで、そのゲームレベルで希望する雰囲気を作り出すことができる。例えば、プレイヤーを正しい方向に誘導したり、注意を引いたり、敵から隠したりする。

サウンド伴奏

さまざまなアクションやイベントで再生されるバックグラウンド・サウンドトラックや効果音。詳しくは「第 4 章 サウンドテスト」を参照。

ゲーム機能

ゲーム機能には、ゲームプレイを特定のレベルで編成するための設定のシステムが含まれる。設定には以下の種類がある：

レベルの進行に関する設定。これらには以下が含まれる：

- 登場人物と敵の出現ポイント
- 勝利と敗北の条件
- ゲームの自動セーブのポイント
- 特定のゲームイベント（スクリプト）を起動するトリガーのシステム。例えば、2 分ごとに戦場上空を飛行機が飛んだり、キャラクターが建物に入るとストーリービデオが流れたり、ガードが保護区域の境界を越えると警報が鳴ったりする

例えば、ドア、拾得物、破壊可能なオブジェクト、トラップなど。

キャラクターがレベルから出たり、ジオメトリにはまり込んだりするのを防ぐ、レベルの物理シェル設定。

原則として、ゲーム機能の設定はプレイヤーからは隠されており、レベルエディタでのみ見ることができる。

5.1.2　レベルデザインにおける欠陥の種類を理解する

レベル作成時に最もよく見られる欠陥がいくつかある。このような欠陥の追跡は困難であるため、しばらくの間、レベルにおいて気づかれずに残ることがある。

専用のマップ作成ツールの中には、マップ上のそのような欠陥を検出する機能が組み込まれているものがある。レベルデザイナーは、しばしばマップやレベルを作成する最終段階でこれらのツールを使用する。しかし、ほとんどの場合、マップの欠陥を発見する最善の方法は、問題を発見し、それをレポートすることができる経験豊富なプレイヤーによってテストされることである。

レベルで発見された欠陥の大部分は、ゲームオブジェクトの外観や位置、照明、コリジョンモデルなどに関連するものである。このような欠陥の詳細な説明は、セクション「3.グラフィックステスト」に記載する。

しかし、以下の欠陥はレベルの設計に直接関係する。

ジオメトリ

ジオメトリの欠陥とは、プレイヤーのキャラクターがレベル/マップの一部で部分的に動かなくなる状況である。原則として、このような欠陥はマップの端、飛び出た棚、斜面で発見される。プレイヤーは、キャラクターをオブジェクトに誘導したり、オブジェクトの近くを歩いたりして、オブジェクトにはまり込んでしまう。多くの場合、キャラクターはジャンプや屈伸、その他の動作の助けを借りて、そのような状況から抜け出すことに成功するが、場合によっては役に立たないこともある。このような場合、プレイヤーはレベル／マップを再スタートするか、最後に進行したセーブポイントに戻るか、特別なボタン／キーの組み合わせを使って、コリジョンの欠陥のない最も近い場所に移動しなければならない。

例えば、ゲームキャラクターがマップのジオメトリにはまり込んで出られなくなったり、アクセスを拒否すべきエリアに入り込んだりすることがある。多人数参加型ゲームでは、このような欠陥を特定することが特に重要である。レベルの欠陥によって、1 人のプレイヤーや 1 つのチームが有利な立場になる可能性がある。

もう 1 つのよくある状況は、動けなくなったときの死、つまりキャラクターがレベル／マップの表面下に視覚的に落下し、下に落ちていくことである。最終的には、キャラクターが死んでレベルをやり直すか、虚空に延々と落下することにより、ゲームを再開しなければならない。

侵入不可能領域

ビデオゲーム、特にマルチプレイヤーゲームのマップでは、プレイヤーが他のプレイヤーよりも有利な立場を得られる場所が意図的に存在することがある。例えば、丘の上の機関銃手には広い射撃エリアがある。このような位置を最初に取ったプレイヤーは、対戦相手よりいくらか有利になるが、これは開発者の意図であり、レベルのデザイン要素であり、欠陥ではない。

しかし、レベルデザイナーの誤りによって、そのような場所が偶然できてしまう状況がある。開発者のコンセプトでは、このエリアはプレイヤーがアクセスできないはずである。しかし、あるゲームキャラクターがそのユニークな特徴（例えば、他のキャラクターよりもジャンプ力が高いなど）を利用して、そのエリアに入ってしまうことがある。こうすることで、プレイヤーはライバルよりも不正にゲームを有利に進めることができる。例えば、他のプレイヤーに発見されない、ダメージを与えられないなど。

複雑なゲームプレイ

ゴールポインターがなかったり、レベルをクリアするために必要なアクションが自明でなかったりすることに関連する問題。例えば、プレイヤーはアクセス可能な領域ですべての敵を倒したが、それ以上進むためには他に何をしなければならないのか理解できない。

ゲームバランスレベル

レベルバランスの設定には多くの欠陥がある。例えば、ゲームの現段階では複雑度が高すぎて、このレベルを通過する可能性を完全に排除してしまうようなボット相手。他のゲームでは、複数のチームが対戦相手より先にポイントマップの目的のポイントに到達しようとする。この場合、レベルバランスの欠陥は、チームにとって不平等なスタート条件かもしれない。

不適切な制限と慣例

これらの問題はゲームプレイには影響せず、ゲームの優位性をもたらすものでもないが、ゲームに対する認識や没入感を低下させる。

例えば、プレイヤーはキャラクターのパスが腰の高さのフェンスに阻まれているのを見るが、それを乗り越えることは不可能である。レベルデザイナーの計画どおり、この場所はプレイヤーが立ち入ることのできない領域であり、そうあるべきである。しかし、プレイヤーから見ると、この状況はゲームへの没入感を台無しにする欠陥に見える。同様に、棒と棒の間の距離が十分にあることはわかるが、キャラクターが鉄格子に閉じ込められて出られない状況を、プレイヤーは欠陥とみなす。

物語

レベルの欠陥は物語の一般的スタイルやゲームプロットの違反に関連するストーリーテリングの一般的なスタイル、ゲームのプロットに対する違反に関連する。このような欠陥がゲームプレイそのものに与える影響は小さいが、ユーザーの没入感やゲームに対する一般的な認識に悪影響を与える。例えば、あるキャラクターが、プロットによれば核爆撃を受けた都市にいることを発見した場合、その都市に無傷の建物が出現すると違和感が生じる。

オブジェクトのカラーコーディングを変更する

ゲーム中にユーザーが遭遇するオブジェクトには、さまざまな特性がある。あるオブジェクトは装飾であり、他のオブジェクトはプレイヤーとのインタラクションが可能である。異なる特性を持つオブジェクトをカラーコーディングすることは、レベルデザインの重要な部分である。従って、一度採用されたカラーコーディングは、ゲーム中に変更されないことが望ましい。

例えば、プレイヤーがそのレベルで赤い樽を発見し、その樽は一発で爆発する。そのような樽に再び遭遇した場合、プレイヤーはその樽から同じ動作を期待するだろう。従って、射撃で爆発するすべての樽は、常に同じ色（通常は赤）でなくてはならない。

プレイヤーが壊すことのできる箱、プレイヤーが開けることのできるドア、プレイヤーがつかまったりよじ登ったりすることのできる岩や棚などは、他のオブジェクトと形や色が異なっているべきである。

そうでなければ、プレイヤーは次に何をすべきかを理解するのに多くの時間を費やさざるを得なくなる。

5.2 ゲームレベルテストの段階と実行

5.2.1 ゲームレベル設計とテストの基本段階

ゲームレベルは、ゲームメカニクス、ビジュアルオブジェクト、サウンドトラック、人工知能、その他のコンポーネントを組み合わせたエンティティである。各コンポーネントのテストアプローチについては、本シラバスの関連セクションで詳しく説明する。

ゲームレベルテストは、これらすべてのコンポーネントをまとめてテストするもので、システムテストに似ており、すべてのコンポーネントが全体として考慮され、テストされる。

ゲームレベルのテストは、その作成の初期段階から開始される。テストの対象、テストの種類、テスト実装の手順、さまざまな専門家の担当領域は、ゲームレベルの現在の開発段階に依存する。

3D ゲームを例に、レベル設計とテストレベルの主な段階を考える。

ゲームレベルのプロトタイピング（デザイナーブロックアウト／グレーボックス）

この段階では、事前に作成したスケッチに基づいて、ゲームの将来のレベルの 3D モデルが作成される。ゲームプレイに直接影響する要素のみが使用される：グレーの立方体、球体、円柱、平面などが、ゲームレベルのレイアウトを大まかに形成する。

このようなモノクロで詳細のない幾何学的な形状のおかげで、レベルの構造を素早く変更することができる。例えば、巨大な場所の断片を削除したり追加したり、必要であれば、すべてをきれいに消してやり直すことができる。この段階では、すべてのゲームメカニクスが使用できる状態になっていないこともあるため、デザイナーは、常に変化する開発状況にレベルを適合させながら、徐々に実装していかなければならない。

レイアウトは、すべての比率とスケールを考慮して作成される。これを行うために、レベルデザイナーはゲームデザイナーと一緒に、ゲームキャラクターが持つサイズとパラメータであるメトリクスのセットを形成する。つまり、ゲームデザイナーは、キャラクターのジャンプの距離と高さ、キャラクターがどのような姿勢になれるか（例えば、完全に立っている、しゃがんでいる、横になっている）と、それ

によって図形のサイズがどのように変化するか、キャラクターが滑らずに移動できる表面の最大傾斜角度などの情報を提供する。これらのメトリックは、オブジェクトやレベル環境の要素の寸法に直接影響する。

レベルデザイナーがレベルのサイズを決定し、オブジェクトの正確なスケールを設定した後、このレベルでのゲームプレイをサポートするために他の機能が設定される。これには、キャラクターの出現ポイント、ゲームイベントの運用性 (トリガーなど) が含まれる。

この段階でのゲームレベルの開発は、原則として何度かの繰り返しで行われ、その間にテストを行うことで、問題のある要素を特定し、取り除くことができる。

ゲームレベルのプロトタイプテスト

プロトタイピングの段階では指定された要件が確定していないため、探索的テストやその他の経験ベースのテスト技法が最も効果的であると考えられている。

作成されたレイアウトでは、このマップでのゲームプレイをテストするために最初のプレイテストが行われる。最初のプレイテストを行うテスト参加者は、ゲームソフト開発チームのテスト担当者でも、外注のテスト担当者でも構わない。テスト参加者からのフィードバックを受けて、開発者は成功した解決策と失敗した解決策について結論を出し、必要に応じてレイアウト上のオブジェクトを変更、移動、または削除する。

また、プロトタイプのプレイテストを行うことで、ゲームメカニクスとテストモデル上のオブジェクトの配置やサイズとの組み合わせがテストされる。

例えば、レベル上にあるシェルターが、キャラクターが敵から隠れるのに十分な大きさであること、表面の隙間の幅がキャラクターが飛び越えられる幅であること、キャラクターが必要なすべての棚をよじ登ることができ、低すぎる天井に頭をぶつけないことなどである。

これとは別に、レベルのレイアウトによって作り出されるゲームの望ましい複雑度と、ゲームプレイ全般から得られる正しい感覚をテストしなければならない。例えば、ゲームのある区間で多くの敵との銃撃戦が想定される場合、レベルには十分な数のシェルターが必要である。1 人の強い敵（ボス）との戦いが想定されているのであれば、レベルの空間は制限され、不必要なオブジェクトは取り除かれる。

ジオメトリ・プロトタイピング（アート・ブロック・アウト／ホワイトボックス）

環境アーティストによって作成された一時的な 3D モデルは、レベルデザイナーがプロトタイプを作成したオブジェクトに取って代わり、さらにオブジェクトが追加される。主なタスクは、ゲームプレイそのものを変更することなく、レベルのビジュアルコンポーネントを調整することである。

しかし、そのような置き換えは、ゲームプレイを複雑にする欠陥をレベルで引き起こす可能性がある。例えば、一時的なツリーモデルがレベルの出力に重なる。この場合、レベルは再びレベルデザイナーの手に渡る。今度は、メカニクスが正しく動作しているかどうか、プレイヤーがどこにでも行けるかどうか、必要なものをすべて見ることができるかどうかを再度テストする必要がある。

ジオメトリ・プロトタイプのテスト

この段階でのテストは、レベルの芸術的な描画の後に、ゲームプレイを妨げる欠陥がないことを確認するために行われる。

そのような欠陥の例としては、レベルの通路を塞ぐ石の仮の 3D モデルや、プレイヤーの視界を遮る樹冠などがある。

このような問題を特定するために、再びプレイテストが行われる。このプレイテストでは、ゲームメカニクスがテストされ、オブジェクトモデルのサイズや位置が必要なメトリクスに対応しているかどうか、ゲームプレイが正しいかどうかがテストされる。

これと並行して、テスト対象の外観、照明、環境などのテストを行うことができる（詳細は「3 グラフィックステスト」に記載する）。

最終バージョンの作成

この段階で、レベルデザイナーは開発チーム全体のソフトウェアを 1 つの全体に統一する。作成されたすべてのコンテンツから 1 つのプレイスペースを収集する。この段階では、レベルに装飾的なディテールが追加され、オブジェクトは、外観、目的、および場所において実際のものと類似するようになる。例えば、机の上には書類や文房具、コンピューターのモニターなどが置かれ、プレイヤーはそれらを叩き割ったり、机から投げ落としたりすることができる。

最終バージョンのテスト

この段階で、性能テストと互換性テストが行われる。さまざまなデバイスでレベルの性能テストが行われ、1 秒あたりのフレーム数が測定され、さまざまな距離、照明、影で対象物の視覚的モデルを表示する品質がテストされる。

プレイテストの一環としてテスト担当者は、すべてのゲームオブジェクトのコリジョンモデルの存在、オブジェクトの物理的モデルと視覚的モデルの対応などをテストする。

例えば、コリジョンモデルのテストは次のように進む。テスト担当者は、ゲームキャラクターをゲーム対象、例えば大きな石に近づけ、さまざまな方向から「石の上を歩く」ことを試みる。石とのコリジョンモデルが適切に設定されていれば、キャラクターの体は視覚的に石に触れるだけで、石を突き破ったり、石の中に完全に入り込んだりすることはない。

5.2.2　関係者の責任範囲

ゲームレベルやマップの作成にはさまざまな専門家が携わるが、多くの場合、これらのタスクは並行して実行されるため、さまざまな欠陥の発生につながる。

品質の高いプロダクトを得るという観点から、これらの専門家が担当する仕事を考える：

役割	責任
ゲームレベルデザイナー	レベルのジオメトリと形状、オブジェクトとトリガーポイントの位置、シェルターのサイズなどを担当する。これらすべてがコアとなるゲームプレイをサポートし、ユーザーのゲームへの関与を促進する必要がある。
アーティスト	芸術的な観点からは、マップ上のすべてのオブジェクトは、正しいテクスチャー、照明、その他の視覚効果を持つべきである。これは見た目（暗い部分と明るい部分の正しい配置）と画像のリアリズムの両方に当てはまる。例えば、森の中にコンクリートブロックでできた小屋があると、奇妙で場違いな印象を与える。
テスト担当者	テスト担当者は、原則として、すべてのオブジェクトがすでに配置され、設定されたレベルやマップの最終バージョンで作業する。テスト担当者の主な仕事は、プレイヤーが与えられたマップで技術的、芸術的な欠陥に遭遇することなくプレイできるかどうかをテストすることである。 マップ上のすべての対象とのインタラクションがテストされる。 ● オブジェクトには物理モデル（コリジョンモデル）がある ● 見えない障害物の欠如、 ● マップから出られないこと

5.3 ゲームレベルのテストをサポートするツール

レベルやマップのテストを容易にするために、3D エディター、レベルエディタ、ビデオゲームエンジンなど、さまざまなツールが使用される。

しかし、テスト担当者はレベルを作成した開発者と同じツールを使用する。

ゲームエンジンは通常、さまざまなテストや欠陥の検索のために、マップやオブジェクトをカスタマイズできる機能を備えている。自社開発エンジンは、テストの有効性を高めるために、テストチームの要求に応じて変更することができる。

例えば、レベル表面のジオメトリの変化の結果として、いくつかのオブジェクトが空中で途切れたり、表面に沈んだりすることがある。このような欠陥は、すべての対象物の外観検査によって発見することができる。また、マップエディターが許可している場合は、特別なアルゴリズムによってテストを簡略化することもできる。

さらに、エディターでさまざまなオブジェクトのテクスチャーの表示を無効にすることで、テスト担当者はマップ上で通行不可能な箇所を発見したり、逆にマップデザイナーが意図していない生存可能なパスを発見したりすることができる。これはマルチプレイヤーゲームでは特に重要である。場合によっては、一部のプレイヤーに他のプレイヤーよりも不本意なアドバンテージを与えてしまう可能性がある。

6. ゲームコントローラーのテスト - 95 分 (K2)

テストキーワード
標準適合性、人間工学テスト、機能テスト

ビデオゲームに特化したキーワード
加速度センサー、ビデオゲームコントローラー、ゲームパッド、ジャイロスコープ、レーシングホイール、 タッチスクリーン、トラックボール

第 6 章の学習目標

6.1 ゲームコントローラーの原理と概念
GaMe-6.1.1 (K2) 典型的な入力デバイスと特殊な入力デバイスを分類する。
GaMe-6.1.2 (K2) さまざまな入力デバイスの例を、そのアプリケーションの観点から挙げる。
GaMe-6.1.3 (K1) さまざまなタイプのゲームコントローラーを想起する。
GaMe-6.1.4 (K2) ゲームコントローラーの仕様に関連したゲーム製品の欠陥とその発生原因を分類する。

6.2 ゲームプロダクトにおけるコントローラーのテストアプローチ
GaMe-6.2.1 (K2) ゲームコントローラーをテストする際のテスト条件の例を挙げる。
GaMe-6.2.2 (K2) ゲームのテストにおいて、UX スペシャリスト、テスト担当者、ゲームデザイナーのタスクを分類する。

6.3 ゲームコントローラーのテストをサポートするツール
GaMe-6.3.1(K2) ゲームコントローラーの動作をテストするためのツールの使用法をまとめる。

6.1 ゲームコントローラーの原理と概念

6.1.1 ゲームコントローラーの種類

ゲームコントローラーは、コンソールゲームやコンピューターゲームで使用される入力装置である。コントローラーは通常、ゲーム機やパソコンに接続されている。ゲームコントローラーを使用して、プレイヤーはゲーム要素の動きやアクションを制御する。この場合、要素の種類はゲーム自体によるが、多くの場合、ゲーム内のキャラクターの 1 つである。

代表的な入力装置

携帯電話、PC、ゲーム機、スロットマシンなどのゲーミングデバイスは、以下のいずれかのデバイスの使用が保証されている。

入力装置	説明
ゲームパッド	ゲーム機の主な入力装置である。
キーボードとマウス	これらのデバイスはパソコン用の一般的な入力デバイスとなっているため、コンピューターゲームにもよく使われている。ゲーム機によっては、マウスやキーボードをゲームに接続して操作できるものもある。

入力装置	説明
レーシングホイール	回転用のノブと 1 つまたは複数のアクションボタンを備えたコントローラーである。
トラックボール	台座からボールが半分飛び出しているように見える。ボールは、上を手のひらでスワイプすることで回転する。
タッチスクリーン	携帯電話、PDA、携帯ゲーム機、最新のスロットマシンで使用されている。ゲームによっては、特にタッチスクリーンをターゲットにしたゲームプレイ要素を持つものもある。

専門機器とは、特定の種類のゲームに特化した機器である。

専用装置	説明
ジョイスティック	もともとは万能ゲーミングデバイスだった。テンポの速いゲームではキーボード/マウスやゲームパッドが好ましいことが発見され、ジョイスティックはフライトシミュレーターのジャンルのゲームに特化したデバイスとなった。しかし、習慣的にゲームパッドを「ジョイスティック」と呼ぶことが多い。
レーシングゲームパッド	ゲーム機でのレースゲームを簡素化するために使用される。実際には、これは通常のゲームパッドに追加のステアリング軸を組み込んだものだ（場合によっては、アナログのスロットルとブレーキボタンもある）。このようなデバイスは、本格的なレーシングホイールよりもはるかに安価である。
操縦桿	民間フライトシミュレーター用（軍用フライトシミュレーターはジョイスティックを使用）。
ペダル	カーシミュレーター向けとフライトシミュレーター向けでは根本的にデザインが異なる。
コンピューター・スロットル	別名エンジンコントロールレバー。フライトシミュレーター用。
シフトレバー	ドライビングシミュレーター用。
ライトピストル	画面上の被写体を撃つ。
グラフィックス・タブレット	マウスの代わりにカーソルを操作するために使用する。
リズムコントローラー	音楽ゲームにおいて、ギター、ドラム、DJ コンソールなどの楽器をシミュレートするために使用される。
ダンス・プラットフォーム	ダンスフロア（スロットマシンのようなもの）。足で踏むことができるいくつかのボタンがついたプラットフォームである。このようなゲームのゲームプレイは、ダンスのように必要な順序でボタンを踏む。

専用装置	説明
ゲーミングキーボード	これは、特定のゲームの仕様に合わせて配置されたボタンと、マクロを作成するための追加ボタンを備えた専用キーボードである。
釣り竿コントローラー	フィッシングシミュレーターゲーム用。
マイク	マイクまたはヘッドセットは、追加の入力デバイスとして使用され、キャラクターとプレイヤーがコミュニケーションできるように、ビデオゲームにコマンドを与える。
列車コントロールパネル	鉄道・路面電車の車両制御用制御盤のシミュレーター。

モーションキャプチャー技術

- 2000 年代初頭から、ヘッドトラッキングシステムはフライトシミュレーターや運動能力の低下した人々のために使用されてきた
- 赤外線センサーと加速度センサーを使って空間内の位置を追跡する遠隔操作装置
- 3D 空間でコントローラーの動きを追跡し、画像を認識するカメラ
- 言葉による命令、体の姿勢、表示された物や絵を使用できる装置

6.1.2 ゲームコントローラーの仕様に関する欠陥について

コントローラーに関連する欠陥の原因は、さまざまである。欠陥の出現は、ソフトウェア自体、コントローラーのコンポーネントの結合、さらには開発者がメーカーからのコントローラーの使用説明書に従わなかったことに起因する場合もある：

- コントローラーのドライバーが古い
- コントローラーモデルとアプリケーションの非互換性
- 個々の装置またはバッチ全体の欠陥
- ゲームプレイと説明書の矛盾

最も一般的な欠陥は、ゲーム中にコントローラーを切り替えたときに置換性がない、またはツールチップがまったく表示されないことである。キーバインドはゲームによって異なる場合がある。また、プレイヤーの判断で割り当てを変更することも可能である。

ドライバーのバージョンが古いか、またはドライバーが存在しない場合、コントローラーが期待どおりに動作しないこともある。ソフトウェアの欠陥がアップデートによって解消できるのであれば、コントローラーの技術的な不具合や欠点は、新しいリビジョンをリリースすることによってのみ修正できる。

しかし、レーシングホイールやその他のコントローラーの動きを読み取る際の不正確さは、ハードウェアの欠陥やソフトウェアの計算の欠陥から生じる可能性がある。ゲームコントローラーの制御信号の読み取り精度が重要なゲームでは、ビデオゲームの設計書に傾斜度による要件値を記載する必要がある。

また、人気のあるプラットフォームでビデオゲームをリリースする場合、プラットフォームの所有者は、そのコントローラーのゲーム内画像の要件を提供することができる。ゲーム内画像とは、ゲーム内で使用されるコントローラーの画像である。例えば、キーバインド設定メニューやゲームパッドキーバインド設定メニューやゲームプレイのヒントで使用される。画像はゲーム内だけでなく、コントローラーをソフトウェアのパッケージやデジタルカバーに表示することもできる。この要件は通常、有名なパブリッシャーや、同時にゲーム機やコントローラーのメーカーに適用される。企業は、ゲーム開発者向けの

テストドキュメンテーションにおいて、アウトラインと商標の両方を含め、アプリケーション内でコントローラーをどのように描写すべきかの要件を提示することがある。

また、コントローラーの加速度センサーやジャイロセンサーを使用するビデオゲームでは、プラットフォーム所有者はセキュリティ要件を課す。最も単純な例は、ゲームプレイを実現するためにコントローラーを持った手を振る必要があることである。コントローラーに内蔵されたモーションセンサーにより、3D 空間でのコントロールとして使用することができる。このような場合、ゲームを開始する前に、ユーザーがコントローラーに取り付けられたホールドストラップを装着する必要があることを指標で示すことが要件となる。そうでなければ、コントローラーで急な動きをすることにより、ユーザーの手から滑り落ち、周囲の機器を傷つけたり、最悪の場合、健康を害したりする可能性がある。

6.2 ゲームプロダクトにおけるコントローラーのテストアプローチ

ゲームコントローラーを使ったテストは通常ゲーム機能がいくつか標準的なコントローラーの使用準備ができた時点から始める。PC の場合はキーボード/マウス、コンソールの場合はゲームパッドを使用する。

このタイプのテストには、以下のようなものがある：

- コントローラーと PC/コンソールの接続/取り外し
- コントローラーのバッテリー残量が少ない
- 特定のコントローラーメーカーへのゲームサポート
- アプリケーションがコントローラーからデータを受け取るための特定の API の互換性
- 1 つまたは複数のコントローラーの使用
- コントローラーの非通常な使用（ネガティブテスト）
- 振動（その有無と強さの度合い）

機能テスト

ソフトウェアが入力デバイスとしてコントローラーの使用を要件とする場合、アプリケーションのすべての機能が、接続されたデバイスと正確に相互作用しなければならない。ユーザーは、コントローラーを変更することなく、ユーザインターフェースの要素とゲームプレイの両方を直接操作できなければならない。割り当てられたコントロール要素：ボタン、スティック、レーシングホイールの回転、音声コマンド、または空間内の動きが、ゲームキャラクターの動作に対応していなければならない。テスト担当者は、可能であれば、コントローラーをデバイスから外したときにアプリケーションが一時停止することも確認する必要がある。ゲームプレイの実行がリアルタイムで行われ、停止することができない場合、ゲームパッドを切断しても、サーバーからプレイヤーが切断されず、同時にインフォメーションメッセージが表示される必要がある。

セキュリティテスト

コントローラーに技術的な脆弱性があり、コンソールをハッキングするためのオープンアクセスがあるかもしれない。セキュリティリスクを軽減するために、セキュリティテストを実施する必要がある：開発者モードへの不正アクセス、フライトモードを使用したネットワーク経由でのデバイスのブロックの回避など。[ISTQB_AL_SEC]。　ゲームコントローラーのセキュリティテストは、ソフトウェアの機能に影響を与えるハードウェアの不具合を発見するアプローチと組み合わさっている。

人間工学テスト

最も人気のあるブランドのコントローラーのボタンには、定着した用途がある。ゲームパッドの X、A や □、X ボタンは、アプリケーション内で、「決定」、「選択を受け入れる」、またはインタラクティブなアイテムとの「対話」のボタンとして、B ボタンと○ボタンは、「キャンセル」、「拒否」、または「戻る」のボタンとして、非常に頻繁に使用される。

ボタンの組み合わせはソフトウェアによって設定され、制御される。ボタンの組み合わせをテストするとき、テスト担当者は、（ジャンル、年齢、障害などによって）異なるグループのプレイヤーにとって

人体構造的に利用できるどうか、プレイヤーが複数のボタンに同時、あるいは順次手が届くかどうかに注意を払う必要がある。

ゲームコントローラーの標準適合性テスト

ゲーム開発において、UI/UXの専門家とゲームデザイナーの役割は必須である。彼らの仕事は、見た目に楽しく、機能的に便利で、一般的に受け入れられている慣習、またはそのジャンルの標準に準拠したインターフェースを開発することである。テストの目的は、コントローラーから送られる入力とインターフェースで実行されるアクションの相互作用の操作性をテストすることと同様に、全体的なUX（ユーザエクスペリエンス）もテストすることである。

テスト担当者は、開発者が一般的なコントローラーの伝統的なレイアウトを使用していることを確認する必要がある。例えば、前述のように、X、A やボタンは「同意」ボタンとして使用され、B、ボタンは「キャンセル」ボタンとして使用される。キーボードの場合は、W、A、S、D ボタンが方向ボタンとして、CtrlまたはCがしゃがんだりもぐったり、スペースバーがジャンプに使われる。マウスの左ボタンの伝統的な機能は、ファーストパーソン・シューターではショットや攻撃、戦略ゲームでは選択となる。マウスの右ボタンは通常、ファーストパーソン・シューターでは照準、戦略ゲームでは部隊の移動に使われる。

テスト担当者はまた、どのコントローラーを使っても、プレイヤーが他のプレイヤーより著しく有利にならないことを確認する必要がある。ゲームパッドはキーボードとマウスのペアに比べ、ターゲットを狙うスピードと精度で大きく劣るため、敵対するターゲットへのオートターゲットとその後の追尾が追加されることが多い。照準は単純に相手に「張り付く」。この場合、照準線がどの距離から相手を追跡し始めるか、また、しばしば弱点となり命中したときに追加ダメージを与える頭部を補足するか、追跡し続けるか、はテスト担当者の責任である。

6.3 ゲームコントローラーのテストサポートツール

録画やコントローラーの表示を図式化することでゲームコントローラーのテストをサポートするいくつかのソフトウェアツールが使用されることがある。

キャプチャー/プレイバックツールは、PCの画面からビデオを録画することができる。これらは欠陥の性質や再現手順を表示するために欠陥レポートに使用される。

別のタイプのツールは、ゲームパッドの表示を図式化しコマンドをビデオに表示するために使用される。これらのツールは、欠陥の発見と解析のために、コントローラーの動作の理解を容易にする。

コントローラーの振動性能だけでなく、コントローラーの入力やゲームパッドのスティックの偏差のテストをサポートするサービスがある。（[URL3]参照）。このようなサービスには、ゲームパッドに必要なテストツールがすべて用意されていてコントローラーの健全性を評価するのに役立つ。

キーボードのキーストロークを表示する特別なツールもあり、テスト実行中に役立つことがある。

7. ローカライゼーションテスト - 155 分 (K3)

テストキーワード
コンプライアンス、国際化、ローカライゼーション

ビデオゲームに特化したキーワード
文化適応性、歴史的正確性、ロケール

第 7 章の学習目標

7.1 ローカライゼーションテストの原理と概念
GaMe-7.1.1 (K1) ローカライゼーションテストの手順を認識する。
GaMe-7.1.2 (K1) 国際化とローカライゼーションの主な目的を想起する。
GaMe-7.1.3 (K2) 国際化とローカライゼーションの機能を比較する。

7.2 ローカライゼーションの欠陥の種類とその原因
GaMe-7.2.1 (K2) ローカライゼーションの欠陥とその原因を分類する。

7.3 ローカライゼーションテストのアプローチと実行
GaMe-7.3.1 (K1) 完全および部分的なローカライゼーションテストを認識する。
GaMe-7.3.2 (K3) ローカライゼーションテストの種類を分類する。
GaMe-7.3.3 (K2) ライター、編集者、翻訳者、ローカライゼーションのテスト担当者のテストタスクを要約する。

7.4 ローカライゼーションテストのサポートツール
GaMe-7.4.1 (K2) ゲームのローカライゼーションテストのためのツールの使用法を要約する。

7.1 ローカライゼーションテストの原理と概念

7.1.1 ローカライゼーションと国際化

ローカライゼーションのプロセスは、それがビデオプレイヤーであれ、ゲーム製品であれ、オペレーティングシステムであれ、同じような段階、アプローチ、テクニックが存在し、ソフトウェア開発会社で採用される独自の内部手順が異なる[Chandler11]。

ゲーム開発とは、コードを書くことでアイデアを実装し、必要なコンテンツを追加してプレイヤーにリリースすることを指す。開発の初期段階で重要なのは、ゲームをリリースする市場を決定することである。

- この段階が必ずしも考慮されるわけではないため、将来的にゲームの売れ行きに悪影響を及ぼす可能性がある
- 国内市場でのプロモーションを成功させるためには、ゲームのローカルバージョンの存在が不可欠である
- ほとんどのゲームは、製品をターゲットの市場に適応させる必要性を考慮して開発される
- ゲームをターゲットの市場に適応させることで、売り上げを何倍にも伸ばすことができる

他市場へのゲームのリリースは、ゲームのローカライズ版の開発終了後に発生する可能性があるため（例えば、会社が他市場でゲームを発売する資金を持っていたり、特定の国で製品を販売促進する準備が整ったパートナーが現れたりした場合）、ゲームを開発する際に、チームはそのゲームをさらに他市

場にも適応できることを考慮する必要がある。そうでなければ、その後の適応に多額の資金やリソースが必要となる可能性がある。

国際化

特定の地域への適応におけるリスクや難しさを避けるために、国際化と呼ばれるプロセスが用いられる。国際化とはほとんどどこでも使用できるように製品を適応させることであるが、ローカライゼーションは特定の地域で使用できるようにするために変化させることである。

ローカライゼーションと異なり、ソフトウェアの国際化とは、その後の翻訳やローカライゼーションを容易にするために開発段階で行われる一連の活動である。

国際化は、開発の初期段階で行われ、ほとんどの場合は言語学者や翻訳者が関与することなく、ソフトウェア開発者の責任で行われる。ローカライゼーションには、多くの追加知識を持つ専門的な翻訳者（場合によってはネイティブスピーカー）が関与する。

国際化は以下を含む。

- ローカライゼーションや国際的な使用に障壁がないようなソフトウェアの作成および開発。あるいはユニコードの使用や文字エンコーディングのアプローチの提供（必要がある場合）
- ローカライゼーションの工程の前には使用できない要素を使用する機会の創出。例えば、マークアップ言語の DTD（Document Type Definition）に双方向テキストのフレームワークを追加する。あるいは、CSS（Cascade Style Sheet）に縦書きテキストや非ラテン文字の基盤を追加する
- 地域的、言語的、文化的な参照をサポートする能力。これには通常、事前に定義されたローカライズされたデータの導入、または以前に作成され、専用のソフトウェアに保存された翻訳物が含まれる。例としては、日付や時刻のフォーマット、ローカルのカレンダー、数値形式や番号体系（ローマ数字とアラビア数字）、リストの選択と表示、個人名と連絡先フォームの使用などが挙げられる
- ローカライズされたバージョンを後でダウンロードしたり、ユーザーの好みに応じて選択したりできるようにするための、コードやコンテンツからローカライズされた要素の抽出。原則として、これはロード可能な言語パックの形式で実装される

このリストには、コンテンツやプログラム、製品の他言語へのローカライゼーションは必ずしも含まれていない。これらは、将来的にローカライゼーションへの移行を容易にするソフトウェア開発のテクニックやアプローチである。

ローカライゼーション

ローカライゼーションとは、ソフトウェアをその国の文化に適応させるプロセスである。ローカライゼーションの特有なケースは、ユーザインターフェース、ドキュメント、関連するソフトウェアファイルの翻訳である [Retsker81]。

ゲームのローカライズは以下を含む。

- ゲームのダイアログや字幕、ツールチップ、説明文およびメッセージ、キャラクターの名前、オブジェクトの名前などの翻訳
- スクリーンセーバー、インターフェース、メニュー項目の翻訳
- テクスチャーやグラフィックの再描画
- 俳優の選定、音声ファイルのアフレコとレコーディング
- ローカライズされた素材のゲームへの統合
- プロジェクトのウェブサイトの翻訳と適応、およびその印刷

- 宣伝材料（ニュース、プレスリリース、マーケティング資料）の翻訳
- ローカライズ版のリリース後のゲームのサポート（アップデート、ニュース、パッチ）

このように、ローカライゼーションのテストは、ゲーム製品が、文化的、言語的、宗教的、政治的、その他の特徴や参照に従って、特定のターゲット層にどれだけ適応しているかをテストすることである。一般的には、ユーザインターフェース、ドキュメント、ファイルの他言語への翻訳、通貨、数字、時間、電話番号などのフォーマットなどは特に、文化的および言語的な側面が考慮される。

ローカライゼーションテストには、ゲームアプリケーションのコンテンツを言語的および文化的要件、そして特定の国や地域の特性に照らしてテストすることが含まれる。このタイプのテストは、最終版の製品がユーザーに届く前に、ローカライズ版の欠陥や翻訳ミスを発見するのに役立つ。ローカライゼーションテストの目的は、異なる市場や地域設定（ロケール）に、異なるローカライズ版の製品の欠陥を発見し、修正することである。

ローカライゼーションは単なる多言語への翻訳ではなく、ローカライゼーションテストと言語テストは同じものではないことには注意が必要である。言語テストは、主にスペル、文法、文体の欠陥のテストで構成されている

7.1.2 ゲームソフトとアプリケーションソフトのローカライゼーションの違い

ゲーム製品とアプリケーションソフトのローカライゼーションのテストの違いは、ローカライズ時に考慮されるさまざまな側面において、この 2 つのタイプの製品が互いに大きく異なるという理解に基づいている。

主な違いは以下のとおりである。

- グラフィックコンテンツのターゲット層への適合
- オーディオコンテンツのローカライゼーション
- テキストコンテンツのローカライゼーションと適合
- ジャンルや文学的特徴の遵守

原則として、大量のグラフィックコンテンツはターゲット層への適合を必要とする場合がある。この種のグラフィックには、標識、キャラクター、ゲームオブジェクト、レベルマップ、記号や装備、ゲームアイテム、広告スクリーンセーバーなどが含まれる。

オーディオコンテンツもまた、慎重なローカライゼーションが必要となる場合がある。多くの場合、ゲーム製品のオーディオコンテンツは、アクターがターゲット層の言語で複製する必要があり、コンテンツ自体も適合させる必要がある。

テキストコンテンツのスタイルが混在している場合は、適切な翻訳と適合が必要である。
ゲームでは次のスタイルを使用する。

- 科学的なスタイル（メカニズムの説明、指示）
- ジャーナリスティックなスタイル（新聞、雑誌、記事）
- 芸術的なスタイル（個人的な日記、本）
- 公務（書類、各種資料）
- 口語的な日常会話（キャラクターたちの会話）

一例として、冒険（クエスト）のジャンルでは、プレイヤーは指示に従っていくつかのデバイスの修理や組み立てを行ったり、新聞で必要な情報を見つけたり、人々に尋ねる必要があるため、科学的、ジャーナリスティック、口語的な日常会話がある。

ゲーム製品には、ターゲット層の歴史的正確性、宗教的、文化的、政治的、イデオロギー的特徴を考慮した適合が義務付けられている。場合によっては、この要件に違反することで、ターゲット層が製品を曖昧に認識し、特定の国や地域での使用が禁止される可能性がある。

ゲーム製品をローカライズする際には、ジャンルや文学的特徴を遵守する必要がある。例えば、マルチプレイヤーRPG の場合、プレイヤーの種族名やオブジェクト名など、そのジャンルで受け入れられている名称を考慮して翻訳する必要がある。ときには、文体的に近く、ジャンルに関連したコンセプトを用いてコンテンツを適合する必要があることもある。

ゲーム製品の高品質な翻訳を実施するためには、追加の知識や情報が必要となる。例えば、ゲームに含まれる可能性のある他のゲーム製品、メディア作品、現実の出来事に関するさまざまな参考文献が挙げられる。

以下は、ローカライゼーションテストの基礎となるローカライゼーションへのアプローチの一部である。

7.1.3　ローカライゼーションテストの段階

ローカライゼーションテストには、翻訳されたコンテンツの正確性、さまざまなインターフェースの要素、欠陥、システムメッセージ、よくある質問とヘルプセクションのテストなどが含まれる。

翻訳テスト

翻訳テストの目的は、ゲームの多言語インターフェースに翻訳の欠陥がないか、住所、姓名、通貨、日付と時刻の形式などが正しいかどうかをテストすることである。

ときには開発中に、数字や数値の体裁を各国の基準（測定単位など）に適応させる必要がある。テストの際、テスト担当者は、速度、長さ、重さ、温度などを明確にユーザーに知らせるために、特定の国でどのシステムが採用されているかを常に覚えておく必要がある。

「あなたは時速 62 マイルで移動している」というメッセージは、メートル単位を使う国のプレイヤーにとっては理解しにくい場合がある。この場合、速度の数値を変えるだけでは不十分で、測定単位も変える必要がある。

上記は、ゲーム内で購入に使用される通貨にも当てはまる。価格のローカライゼーションの最も単純な例は、製品がアクティベートされた地域の通貨に自動的に変換されることである。為替レートでの変換に加えて、対応する通貨記号も存在する必要がある。しかし、アプリケーションを開発するとき、テスト担当者は、ソフトウェア配布サービスが地域価格を設定するかもしれないという事実も考慮しなければならない。

さまざまなビデオゲームでは、アプリ内ストアの価格は通貨に換算され、地域によって異なる。

ゲームのテストでは、グラフィックに注意を払う必要がある。ゲームが公開されている国の現実に対応していなければならない。例えば、道路標識は国によって見た目が異なることがある。また、現地の祝祭日の写真や風景も追加される。イスラム諸国では、人物や動物の絵はすべて削除され、アラベスク（幾何学模様と植物の要素からなる東洋の中世の複雑な装飾）が追加されるなど、グラフィックが根本的に見直される。

中国の旧正月のお祝いは特に人気がある。多くのゲームでは、インターフェースがランタン、龍、花火などの国のシンボルで飾られている。また、ゲームに限定コンテンツ（期間限定で利用可能）が補充されることもある。様式化された中国の神話の英雄のキャラクタースキンや、爆竹やロケットを発射したり、豚や龍に乗ったりすることがそれにあたる。

ゲームでは、ジョークを適応されなければならないことが多く、ときにはプロットさえも、ビデオゲームが販売される国の精神に合わせて調整しなければならない。

ローカライゼーションは、テキストの取り扱いに限らず、文化的特性や道徳的および倫理的側面の調和も含まれる。

従来、ローカライゼーションのテストには、テスト担当者が最も注意を払うべきトピックや方針がいくつかある。

- オカルトや悪魔崇拝を含む宗教
- 性、露出度の高い服装の登場人物、下品な言葉遣い
- 文化や人々そのものにまつわる偏見や固定観念
- 戦争、軍事衝突、テロ
- 歴史に関する異なる国の特定の見解を含む政治

例えば、ビデオゲームの音声にコーランの一節が含まれていた場合、ほとんどのイスラム諸国はこのビデオゲームを禁止する可能性が高い。

ポリティカル・コレクトネスが厳しく統制されている現代社会では、その遵守は真剣に受け止められなければならない。

ゲームの技術的要素や、ゲームのローカライゼーションの準備不足に起因する問題に加え、ビデオゲーム翻訳者は、翻訳するゲームに関連した困難ではなく、高品質の翻訳を行うための追加知識の必要性に関連した数多くの困難に直面することがある。

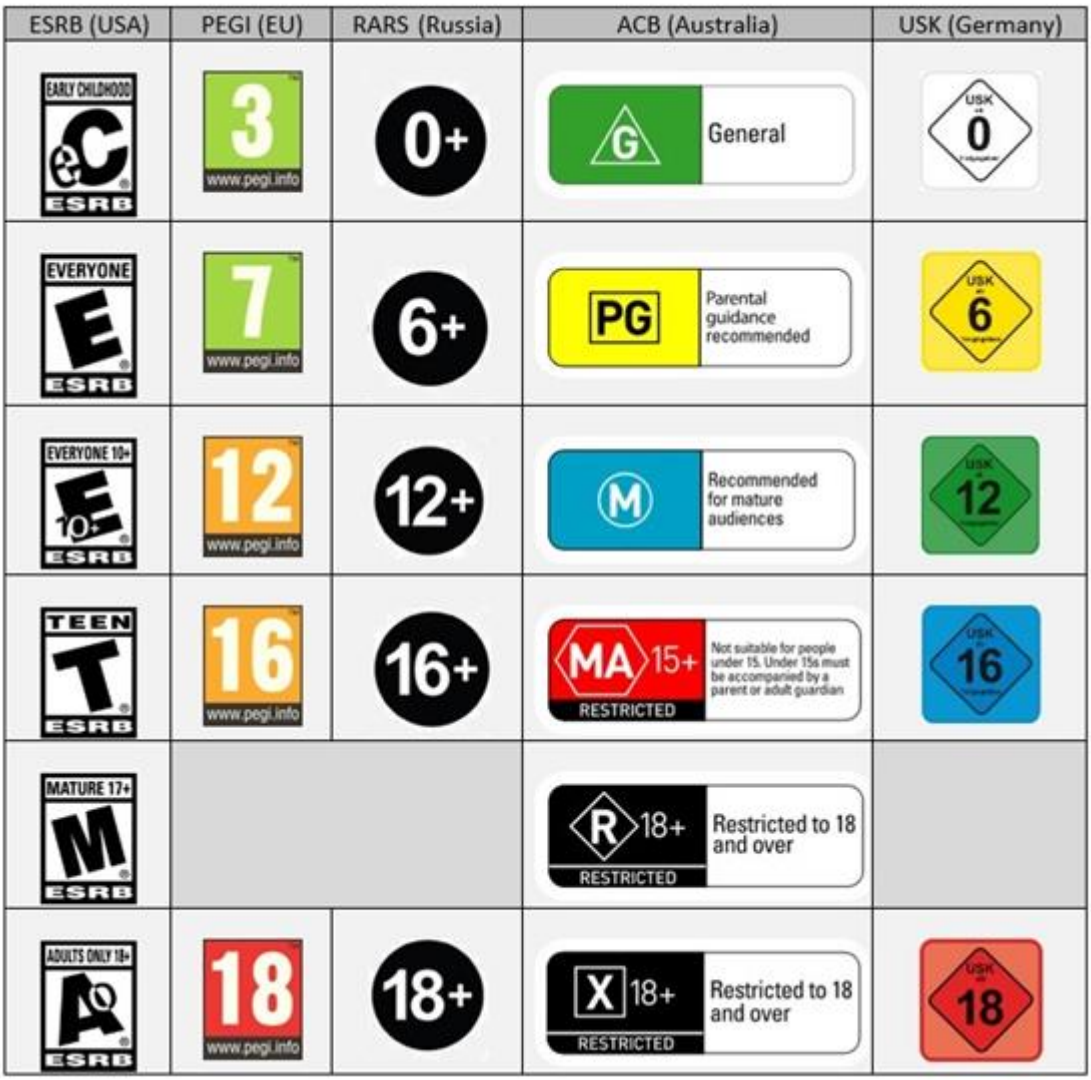
追加知識には、ゲームに含まれる可能性のある他のゲーム製品の背景、メディア作品、現実の出来事も含まれる。このような参考文献は、以前に翻訳された方針に従って適切に翻訳されなければならない。そのため、翻訳者は世界で起きている最新の出来事や、さまざまな人気映画やゲームを認識している必要がある。

コンプライアンステスト

ゲームやアプリケーションの年齢制限システムは、各国の法律や文化の特殊性を考慮している。これにより、開発者はコンテンツの制限をより正確に定義し、対象となる層にアプリケーションを配布することができる。

ローカライゼーションのテストでは、さまざまな地域の法的要件を考慮する必要がある。例えば、ロシアでは成人年齢は 18 歳、アメリカでは州によって 18 歳から 21 歳、日本では 18 歳から成人とみなされる。多くの場合、内容の異なるゲームは、各国で施行されている年齢制限システムに従わなければならない。

同じゲームでも国によって対象年齢が異なる場合がある。



各国のゲームレーティング
https://www.kaspersky.com/blog/gaming-age-ratings/11647/

法律や法的要件を遵守するためのローカライゼーションをテストする場合、多くの状況を考慮する必要がある。場合によってはターゲットの言語ではなく、ゲーム製品が存在する市場の国の法的および文化的特徴に関連することもある。

加えて、各国の通貨とそれを使った運営のゲーム内でのサポートにも注意を払う価値がある。例えば、ある国で外国為替取引がその国の銀行以外には禁止されている場合、その市場でゲームを発売しようとすると、ゲームパブリッシャーは必ず深刻な問題に直面することになる。

間接的な通貨取引としては、ルートボックス（文字どおり「賞品が入った箱」）の購入があり、これには原則として、さまざまなキャラクタースキン、消費アイテム、アップグレード、マウントなどが入っている。

2016 年 12 月、中華人民共和国文化部は、2017 年 5 月よりオンラインゲームのパブリッシャーに対し、すべての仮想アイテムおよびサービスの取得確率を公表することを義務付ける法律の制定を発表した。

オーストラリアでは、ルートボックスゲームが金銭または価値のあるものでプレイできる場合、賭博規制の対象となる。ゲーム内にのみ存在するオブジェクトの価値が、そのオブジェクトの名声に関連してのみ判断できる場合であっても規制の対象となりうる。

色と記号

国によって意味が異なる特定の色や記号も、ローカライゼーションのテストでは考慮する必要がある。

例えば、中国の住民にとって赤は忍耐と信仰の象徴であり、インドでは純潔の象徴である。一方、ヨーロッパではこの色に罪と犠牲を見る。南アフリカの人々にとっては悲しみの色である。アメリカと日本では、赤は危険とテロリストの脅威を象徴し、エジプト人は喪を連想する。従って、ゲームプロジェクトをテストするとき、配色が重要になる可能性があるため、テスト担当者は配色に注意を払う必要がある。

さらに、テスト担当者はテストを行う際、地域ごとのキーボードレイアウトやホットキーのサポートに注意を払う必要がある。

アプリケーションがサードパーティのリソース（例えばクラウドストレージやソーシャルネットワークなど）との統合を使用する場合、その地域での可用性を考慮する必要がある。

従って、ローカライゼーションと国際化のテストプロセスは異なる：

国際化	ローカライゼーション
UTF エンコーディング	翻訳
データフォーマット	法的要件
文章の方向	通貨および外国為替取引
	色と記号
	キーボードレイアウトとホットキー
	サードパーティのリソースとの統合

7.2 ローカライゼーションの欠陥の種類とその原因

7.2.1 ゲーム製品のローカライゼーションにおける欠陥の考えられる原因

大企業でさえ「文化的」な欠陥から免れることはできない。どのようなアプリケーションをリリースする場合でも、テスト担当者は前例や既存の規制に精通する必要がある。場合によっては、1 つのフレーズやジョークが、特定の国でのゲームの完全な販売禁止につながることもある。ローカライゼーションの欠陥（特定の地域で禁止されている内容やコンテンツなど）の原因としては、以下のようなものが考えられる。

- 画像
- サウンドやサウンドトラック
- 現実的な、あるいは歴史的なシーン
- フレーズや引用

7.2.2 ローカライゼーションの欠陥とリスク

ターゲットの言語や製品自体の知識が不十分または不足していると、特に製品がシミュレーションのジャンルを代表する医療、技術、またはスポーツである場合、ローカライゼーションテストが大幅に複雑になる可能性がある [URL4]。

また、さまざまな地域の特性を調査するのに時間がかかるため、ローカライゼーションのテストはかなり長いプロセスになる可能性があることにも留意する必要がある。

主なローカライゼーションの欠陥は以下のように分類できる。

技術的側面

- 文字化け（意味不明な文章） / ゴミの文字。間違って選択された文字エンコーディングを使用してテキストをデコードした結果、歪んだテキストが表示される。その結果、多くの場合は異なる書記体系のまったく関係のない文字に規則的に置換される。通常、これによりテキストが読めなくなる
- ローカライズされた文字列がインターフェースで設定された境界に収まらない（切り取られたり、スクロールされたりする）。ある言語から別の言語に翻訳された用語は、異なる文字数を使用する場合がある
- オフセット。アプリケーションをローカライズした後、元の配置を維持するためユーザインターフェースの要素のレイアウトの再構成が必要になる場合がある
- ローカライズされた言語に一節ごと翻訳がない。テスト担当者は、ダイアログボックスのテキスト、ドキュメントやユーザインターフェースの画像やスクリーンショットに注意を払う必要がある。ユーザーの期待に応えるためには、これらのコンテンツをすべてローカライズする必要がある
- オーバーラップ。これはウィンドウやダイアログ内の特定のコントロールが他のコントロールと重なる場合に起こる
- テキストの欠落。アプリケーションの翻訳や作成中に、テキストが失われることがある
- フォントやサイズの間違い。国によってデフォルトのフォントやサイズは異なる。例えば、アジア諸国では通常、デフォルトでフォントサイズ 9 が使用されるが、アメリカではフォントサイズ 8 が使用される。この問題は通常、機能に支障をきたすことはないが、テキストが読みづらくなりユーザエクスペリエンスに大きな影響を与える
- 間違ったホットキー。ローカライズされた言語またはキーボードにその文字がないため、ホットキーが使用できない場合がある

- テキストのテンプレートの変数。異なるタイプの言語（分析的言語や総合的言語）では、異なる形の変化、格、複数形が使われ、語尾や音を変えたり変えなかったりするため、変数が単語の形に影響を与えることがある。開発者が犯しがちな一般的な間違いの 1 つに、変数を含む文を分割してしまうことがある。ターゲットの言語の文法構造は、ローカライゼーションのほぼ主要な基礎である。実際、変数を追加するとき、多くの言語では、語句の構造によりその意味が変わる可能性があることを開発者が考慮していないことが多々ある
- 原文と訳文の音の順序のずれ。翻訳する際、音節数の違いにより登場人物のフレーズが原文より長くなったり、逆に短くなったりすることがある。ゲームのビデオクリップは事前に録音することが多いため、ローカライザーはキューの翻訳に対してより慎重なアプローチを取らざるを得ない。翻訳精度と音節数のバランスを保ちつつ、キャラクターの唇の動きと音の同期（リップシンク）を考慮することが重要である

翻訳の欠陥

- 固有名詞、日付、数値、歴史的出来事の名称、祝日、現実、専門用語、冒涜的表現、口語的語彙、略語、「連想させる」名称、ニックネームなどの誤った翻訳または音訳
- 矛盾した用語。同じ用語は、アプリケーション全体を通して一貫した訳語であるべきである。翻訳に互換性がないことはよくある

コンテンツの文化適応性

- ユーモア。ジョーク、ユーモア、風刺の対象に対する態度や個々の対象との関係におけるユーモアとその許容性の考察など
- 宗教。宗教的な問題に対するターゲット層の態度。宗教的な場所、儀式、式典を含む。少数のカルト（悪魔崇拝や邪教）に対するターゲット層の態度
- 歴史の正確さと出来事の認識。歴史上の出来事、軍事衝突、科学分野の発見、現存する特定の物体の外観や特徴、人々の生活様式の特殊性などの解釈
- 国の文化や世界観の特殊性（子ども、性、暴力、異文化に対する態度などを含む）。国の固定観念、国の料理、服装、人々のライフスタイル。子どもに対する態度、性的少数者に対する態度、動物に対する暴力など
- 法的規制。年齢制限、児童保護分野の法律、宗教的信条、暴力のプロパガンダ、薬物、性的関係、人種、テロリズム、政府や国家制度に対する言論、特定のシンボルの使用禁止などを考慮する
- 過剰なローカライゼーション。すべてをローカライズする必要はなく、例えば、商標、ロゴ、略語、商品名など一部の要素は翻訳せずに元の外観を保持する必要がある

7.3 ローカライゼーションテストのアプローチと実行

7.3.1 完全ローカライゼーションテストと部分ローカライゼーションテストの違い

完全ローカライゼーションテスト

これは、欠陥に対するローカライゼーションの徹底的なテストを意味する。このテストタイプは、クライアントの新しいローカライゼーションを開発する場合にのみ有効で、すべての文字列が翻訳され、事前にテストされていないことが条件となる。あらゆるタイプのテストが含まれるため、最も費用がかかる。

部分ローカライゼーションテスト

これは、以前にテストされたローカライゼーション内でテキストが変更された場合に使用される。検証は、メインストーリーとそれを参照するローカライゼーションに影響を与えるメジャーアップデートの一部として変更された行を特定する。テストされるのは、変更されたテキストのみであり、そのテキス

トは以前にゲーム内で提示されたテキストと一致していなければならない。このテストアプローチは、コストを最小限に抑えながら効率性を達成するという点で最適なものと考えられる。

7.3.2 ゲーム製品のソフトウェア開発ライフサイクルにおけるローカライゼーションのテスト手順とアプローチ

ローカライゼーションテストでは、翻訳、サポートファイル、インターフェース要素の正しい配置と適応、およびテキストの記述ルールを検証する。

ローカライゼーションテストのゴールは、ゲームが多言語のインターフェースと機能性をサポートしていること、およびローカライゼーションの問題（別の言語への翻訳、日付と数字の形式、郵送先住所、姓と名の順序、通貨など）がないことを確認することである。

ローカライゼーションテストプロセスには以下のようなものがある。

- 対応言語リストの決定と検討
- ユーザインターフェース要素、システムメッセージ、欠陥など、翻訳の正確性のテスト
- 「ヘルプ」セクションおよび付属文書（存在する場合）の翻訳の検証

ローカライゼーションテストでは、ローカライゼーションチームが翻訳した文字列を、参照のローカライゼーションの文字列と比較して次のことを確認する。

- 文法、句読点、構文の欠陥
- テストされたローカライゼーションの地域要件への違反（時間や日付や単位の形式、法令遵守など）
- 必要な技術データの欠如（変数、テキストの書式に使用されるセクションなど）
- 参考のローカライゼーションのアートスタイルとコンテキストの違反
- ゲームインターフェースのテキスト表示の欠陥（長すぎる行、フォントの表示形式違反など）

上記のテストはすべてテストチームが実施しなければならない。ただし、ゲームが翻訳された各言語の知識を持つテスト担当者を見つけることは非常に困難なため、参照のローカライゼーションの文法、句読点、構文、アートスタイルやコンテキストをテストできるとは限らない。これらのテストの実行はローカライズ担当者側で行われる。テスト担当者は、テスト対象のローカライゼーションの地域要件への違反をチェックし、必要な技術データの不足をチェックし、ゲームインターフェースのテキスト表示の欠陥を特定する。（詳細は「7.3.3. ローカライゼーションテストの種類」を参照）

ローカライゼーションテストの前段階

この段階には以下が含まれる。

- テスト担当者に必要なすべての製品ドキュメントの提供
- テスト担当者が使用されている用語を正しく解釈できるための用語集と翻訳記録の作成
- すでにローカライズされている場合に評価目的での旧バージョンの提供
- 欠陥管理ツールの選択と構成 - ローカライゼーションテスト中に発見されたすべての欠陥が記録される文書またはプラットフォーム

地域的および文化的特性のテスト

これはローカライゼーションテストにおいて最も重要なステップの 1 つである。スクリーンショットまたはローカライズされたビルドが使用される。以下をテストする必要がある。

- 選択した地域の日付と時刻の形式
- 電話番号と住所の形式

- 配色
- 製品名の地域標準への遵守
- 通貨形式
- 単位

言語テスト

言語機能がテストされる。テスト担当者は以下のことを確認する必要がある。

- 同じ用語が使われている
- 文法的な欠陥がない
- スペル間違いがない
- 句読点が規則に従っている
- テキストの方向が正しい（右から左、または左から右）
- ブランド名、都市名、地名、位置などが正しく設定されている

ユーザインターフェース（または外観）

以下のことを確認することが重要である。

- 画像のキャプションがすべてローカライズされている
- ローカライズ版のレイアウトがオリジナルと同じである
- ページや画面上の改行が対象の言語のルールに従って配置されている
- 会話、ポップアップ、通知が正しく表示される
- 行の長さが既存の制限を超えず、テキストが正しく表示される（翻訳テキストが原文より長くなりボタンに収まらない場合がある）

機能性

ローカライズされたアプリケーションが正しく動作するかをテストする必要があるため次の点に注意する。

- ローカライズされた製品の機能性
- 情報を入力する機能
- 異なるロケールや言語の特殊文字のサポート
- キーボードショートカットのサポート
- さまざまなフォントのサポート
- さまざまな形式の区切り文字のサポート

ビデオテストとスケールテストに含まれるもの

- スケールの長さとそれが属するゲームの要素の対応
- ジェンダー別のキャラクターのスケールの対応
- 音の純度（干渉がないことや元の音源と翻訳された音源のスケールの同期と相対的な音量）
- 歴史的な側面も含めたサウンドトラックの信憑性
- 音の文体的および文化的特徴（アクセント、話し方の特徴）

7.3.3　ローカライゼーションテストの種類

「ボックス」ローカライゼーション

ゲームが物理的なメディアで発売および販売される場合、パッケージに書かれている内容がローカライズされる。物理的な媒体ではなくプラットフォームで販売される場合は、ストアのページの説明文とスクリーンショットが翻訳される。ボックスのローカライゼーションはこれに限定される。

インターフェースのローカライゼーション

説明文とボックスは、ゲーム、インターフェース、ヘルプページ、ボタンのラベルなどで翻訳される。プロットが完全に別の言語であるが、"Play"ボタンの表記が 1 つの言語で作られている場合、珍しいタイプのローカライゼーションが発生する。

テキストのローカライゼーション

通常、ゲーム内のテキストは字幕に至るまですべて翻訳される。つまりユーザーは、例としてゲーム中のアフリカ系アメリカ人のスラングを聞いて理解しようとしても、他の言語の字幕を見ることになる。

声優によるローカライゼーション

発言やセリフは翻訳され、声優が声をあてる。声優によるローカライゼーションが適切なレベルで行われていれば、それが外国のものとして認識されることはない。

グラフィックのローカライゼーション

どんなゲームにも、何らかのエンジン、デザイン、グラフィックオブジェクト、テクスチャーが含まれる。例えば、ゲーム内のフェンスの題字などテキスト以外のものはすべて含まれる。グラフィックのローカライゼーションとは、その中にあるすべての碑文を翻訳しなければならないことを意味する。これには新聞、店の看板、メモなどが含まれる。

多くの場合、ゲーム中のアクションは特定の場所で行われる。同時にローカライゼーションへのアプローチも異なってくる。あるオブジェクト（例えば新聞のメモなど）がプロットに役立つ場合、それらを翻訳する必要がある。そうでなければ、重要なポイントが失われる。同時に、ゲームが行われる場所の言語で書かれた壁の碑文は、単なる飾りであれば翻訳する必要はない。

深いローカライゼーション - 文化的適応

ゲームを完全に作り直す際、これは文化的適応である。残るのはソースコードとメカニクスだけになる。開発者はテクスチャー、プロット、セリフ、キャラクターモデルを作り直し、ビデオゲームエンジン上でまったく別のゲームを作ることができる。このようなことはめったに行われないが、この方法でのローカライゼーションは今でも行われている。これは、このような適応がないとゲームが対象者に使われず、特定の市場で販売できない場合に行われる。

7.3.4　関係者の責任範囲

役割	責任
ライター / 物語デザイナー	● ローカライゼーションの戦略と計画の開発 ● 用語の定義、ゲームジャンルに特有の概念や、プロジェクトに関連する翻訳が難しい用語の翻訳 ● 編集者や翻訳者への文脈の説明
翻訳者	● テキストの翻訳

役割	責任
編集者	● 翻訳された資料の校正 ● ローカライゼーションが行われる言語の道徳的および倫理的側面とともに文化的特徴を遵守しているかのテスト ● 翻訳スタイルの統一性の制御
ローカライゼーションテスト担当者	● 翻訳の正確性のテスト（発言の文脈と意味、テキストの一貫性、プレイスタイルへの遵守） ● 翻訳の技術的な遵守のテスト（外観、インターフェース、フォントの機能、区切り文字、特殊文字など） ● （内外のアプリケーションとの）翻訳の技術的結合の制御

7.4 ローカライゼーションテストのサポートツール

ローカライゼーションテストのツールは、テストのさまざまな段階で用いられ、さまざまな問題に対処することができる。これらのツールには以下のようなものがある：

- 視覚的な文字列比較ツール。これらのツールは、参照の文字列とターゲットのローカライゼーションの文字列を比較するのに役立つ
- 自動文字列比較ツール。このタイプのツールには、補助的なスクリプト、プログラム、ユーティリティが含まれ、その結果としてテストされたローカライゼーションにおけるあらゆる種類の欠陥の存在に関するデータを取得することができる
- ローカライゼーションのファイル構造をテストして、欠落しているファイルや冗長なファイルがないかをテストするツール
- ローカライゼーションをテストするための自動化ツールの例として次のようなものがある。
    - 2 つのローカライゼーション（参照と検証済）のスクリーンショットを比較し、重大な不一致があればエラーを生成するツール
    - 参照内の変更の存在に基づいてターゲットのローカライゼーション内の変更の存在を判断するツール
    - 対になっている参照とターゲットのローカライゼーションの文字列の変数と数値を比較するツール

8. 参考文献

8.1 標準

[ISO25000] ISO/IEC 25000:2014, Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE
JSTQB 訳注) 日本での標準は　JIS X 25000：2017(ISO/IEC 25000：2014)
システム及びソフトウェア製品の品質要求及び評価（SQuaRE）－SQuaRE の指針

[ISO25010] ISO/IEC 25010:2011, Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models
JSTQB 訳注) 日本での標準は　JIS X 25010：2013 (ISO/IEC 25010：2011)
システム及びソフトウェア製品の品質要求及び評価（SQuaRE）−システム及びソフトウェア品質モデル

8.2 ISTQB ドキュメント

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

[ISTQB_ALTA_SYL] ISTQB Advanced Level Test Analyst Syllabus, Version 3.1.2
JSTQB 訳注)　Advanced Level シラバス日本語版 テストアナリスト Version 3.1.1.J03

[ISTQB_ALTTA_SYL] ISTQB Advanced Level Technical Test Analyst Syllabus, Version 4.0
JSTQB 訳注)　Advanced Level シラバス日本語版　テクニカルテストアナリスト Version 2012.J02

[ISTQB_ALTM_SYL] ISTQB Advanced Level Test Manager Syllabus, Version 2012
JSTQB 訳注)　Advanced Level シラバス日本語版　テストマネージャ Version 2012.J04

[ISTQB_CTFL_MAT] ISTQB Mobile Application Tester, Version 2019
JSTQB 訳注)　Foundation Level Specialist シラバス モバイルアプリケーションテスト担当者 Version 2019.J01

[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
JSTQB 訳注)　Foundation Level Extension シラバスアジャイルテスト担当者 Version 2014.J02

[ISTQB_FL_PT] ISTQB Foundation Level Performance Testing Syllabus, Version 2018
JSTQB 訳注)　Foundation Level Specialist シラバス 性能テスト担当者 Version 2018.J01

[ISTQB_FL_SYL] ISTQB Foundation Level (Core) Syllabus, Version 2018
JSTQB 訳注)　Foundation Level シラバス　Version 2018V3.1.J03

[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

8.3 書籍

[Nystrom14] Nystrom, R. (2014). Game programming patterns. Genever Benning., ISBN: 978-0990582908. URL: https://www.gameprogrammingpatterns.com/
JSTQB 訳注)　邦訳版 Game Programming Patterns ソフトウェア開発の問題解決メニュー

Robert Nystrom(著), 武舎広幸(監訳), 阿部和也,上西昌弘(訳)
出版社：インプレスブックス
ISBN　9784844338901
https://book.impress.co.jp/books/1114101121
https://tatsu-zine.com/books/game-programming-patterns

[Gregory18] Gregory, J. (2018). Game engine architecture. AK Peters/CRC Press., ISBN: 978-1466560017. URL: https://www.gameenginebook.com/
JSTQB 訳注)　邦訳版
ゲームエンジンアーキテクチャ 第 3 版 単行本 – 2020/12/13
ジェイソン・グレゴリー (著), 今給黎 隆 (監修), 湊 和久 (監修), 加藤 諒 (編集)
出版社：ボーンデジタル
ISBN　978-4-86246-483-5
https://www.borndigital.co.jp/book/19115/

[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).

8.4 リンク (ウェブ/インターネット)

注：すべての参考文献は 2022 年 4 月 28 日現在のものである。

- [ISTQB-Web] https://www.istqb.org/
  JSTQB 訳注）　JSTQB-Web は　https://jstqb.jp/index.html
- [URL1] https://research.ncl.ac.uk/game/mastersdegree/workshops/technicalrequirementschecklists/Technical%20Requirements%20Checklist%20Workshop.pdf
- [URL2] - JSTQB 訳注）リンク切れ
- [URL3] https://gamepad-tester.com/
- [URL4] - JSTQB 訳注）リンク切れ

9. 付録 A - 学習目標／認知的知識レベル

本シラバスに適用される学習目標は以下のとおりである。シラバスの各トピックは、その学習目標に従って試験される。

学習目標は、以下のように、知識の認知レベルに対応する動作動詞で始まる。

レベル 1：記憶レベル（K1）

用語または概念を認識し、記憶して、想起することができる。

動作動詞：想起する（Recall）、認識する（recognize）。

例
テストピラミッドの概念を想起する。
テストの典型的な目的を認識する。

レベル 2：理解レベル（K2）

トピックに関連する記述について理由または説明を選択することができ、テスト概念に関して要約、比較、分類、例の提示を行うことができる。

動作動詞：分類する、比較する、区別する、区別する、説明する、例を挙げる、解釈する、要約する

例	備考
テストツールを、その目的とサポートするテスト活動に従って分類する。	
異なるテストレベルを比較する。	類似点、相違点、またはその両方を探すのに使用できる。
テストとデバッグを識別する。	コンセプトの違いを探す。
プロジェクトリスクとプロダクトリスクを区別する。	2 つ（またはそれ以上）の概念を別々に分類できるようにする。
テストプロセスにおけるコンテキストの影響を説明する。	
テストが必要な理由を例示する。	
与えられた故障プロファイルから欠陥の根本原因を推測する。	
作業成果物レビュープロセスの活動を要約する。	

レベル 3：応用（K3）

身近な課題に直面したときに手順を実行できる、または正しい手順を選択し、与えられた状況に適用できる。

行動動詞：適用する、実施する、準備する、使用する

例	備考
境界値分析（boundary value analysis）を適用し、与えられた要件からテストケースを導き出す。	手順／技術／プロセスなどを指すこと。
技術要件と管理要件をサポートするメトリック収集方法を導入する。	
モバイルアプリの設置性テストを準備する。	
トレーサビリティを利用して、テスト目的、テスト戦略、テスト計画書との整合性が完全であるか、テストの進行状況をモニタリングする。	受験者がテクニックや手順を使用できることを求めるLOで使用できる。「apply」に似ている。

10. 付録 B-学習目標とビジネス成果のトレーサビリティマトリクス

このセクションでは、「認定テスターによるゲームテストのビジネス成果」と「認定テスターによるゲームテストの学習目標」のトレーサビリティを示す。

ビジネスの成果認定テスト担当者 ゲームテスト	GaMe-1	GaMe-2	GaMe-3	GaMe-4	GaMe-5	GaMe-6
GaMe-1 ビデオゲームとゲームテストの基本概念を説明できる。	17					
GaMe-2 利害関係者のニーズと期待に従って、リスク、目標、ゲームソフトウェアの要件を決定する。		12				
GaMe-3 基本的なゲームソフトウェアのテストを概念的に設計、実装、実行できる。			6			
GaMe-4 ゲームソフトウェアのテストアプローチとその目的を知る。				9		
GaMe-5 ゲームテストをサポートするツールを認識する。					7	
GaMe-6 テスト活動をソフトウェア開発ライフサイクルとどのように整合させ、ビデオゲームの開発とパブリッシングのコストを削減するかを判断する。						6

			GaMe-1	GaMe-2	GaMe-3	GaMe-4	GaMe-5	GaMe-6
1.ゲームテストの特異性								
GaMe-1.1.1	K1	ゲームテストの目的と特徴を理解する。	x					
GaMe-1.1.2	K2	ゲームソフトのプロダクトリスクについて例を挙げる。	x	x				
GaMe-1.1.3	K2	ゲームテストに関する欠陥について例を挙げる。	x					
GaMe-1.1.4	K2	ゲームテストのリスクを軽減する方法を要約する。		x				
GaMe-1.1.5	K2	ゲームのテストとプレイを比較する。	x					
GaMe-1.2.1	K1	ゲーム開発チームにおける具体的な役割とテスト作業を識別する。	x					
GaMe-1.3.1	K1	ゲームソフトウェア開発ライフサイクルを通してテスト活動を想起する。					x	
2.ゲームメカニクスのテスト								
GaMe-2.1.1	K2	ゲームの仕組みの種類を分類する。	x					
GaMe-2.1.2	K2	ゲームプレイメカニクスのテストと非ゲームプレイメカニクスのテストを区別する。			x			

GaMe-2.1.3	K2	コアメカニクスとメタメカニクスのテストを区別する。			x			
GaMe-2.1.4	K2	クライアント、サーバー、クライアントサーバーメカニクスのテストを区別する。			x			
GaMe-2.1.5	K2	ゲームメカニクスの欠陥の例を挙げる。		x				
GaMe-2.2.1	K2	ゲーム制作のさまざまな段階における主なアプローチとテスト対象を要約する。				x	x	
GaMe-2.2.2	K2	ゲームメカニクスをテストすることの重要性を理解する。	x					
GaMe-2.2.3	K2	ゲームメカニクスを説明した文書をレビューすることの重要性を理解する。	x					
GaMe-2.2.4	K3	ゲームメカニクスのテストの基本的なアプローチを適用する。				x		
3.グラフィックステスト								
GaMe-3.1.1	K2	ゲームプロダクトのグラフィックコンテンツの特徴を説明する。	x					
GaMe-3.1.2	K2	グラフィックコンテンツの欠陥の種類を分類する。		x				
GaMe-3.2.1	K2	アーティスティックテストの主なアプローチを要約する。				x		
GaMe-3.2.2	K2	テクニカルテストの主なアプローチを要約する。				x		
GaMe-3.2.3	K2	ゲームプレイテストの主なアプローチを要約する。				x		
GaMe-3.3.1	K3	グラフィックステストの基本的なアプローチを適用する。			x			
GaMe-3.3.2	K2	グラフィックスの歴史的な妥当性をテストすることの重要性を説明する。		x				
GaMe-3.4.1	K2	グラフィックステストのツールの使用法を要約する。					x	
4.サウンドテスト								
GaMe-4.1.1	K1	ゲームプロダクトのサウンドコンテンツの特色を想起する。	x					
GaMe-4.2.1	K1	サウンドコンテンツの欠陥の種類を想起する。		x				
GaMe-4.2.2	K2	サウンドコンテンツの欠陥を分類する。		x				
GaMe-4.3.1	K2	オーディオコンテンツのテストへの主なアプローチを要約する。				x		
GaMe-4.3.2	K2	音楽とサウンドのミックスをテストするための主なアプローチを要約する。				x		
GaMe-4.3.3	K2	作曲テストの主なアプローチを要約する。				x		
GaMe-4.4.1	K2	音楽コンテンツのテストレベルについて説明する。	x					
GaMe-4.4.2	K1	サウンドをクライアントに統合することの特色を想起する。	x					
GaMe-4.4.3	K1	サウンドテストの責任範囲を想起する。						x

GaMe-4.4.4	K3	サウンドテストへのアプローチを適用する。			x			
GaMe-4.5.1	K2	サウンドテストツールの使用法を要約する。					x	
5.ゲームレベルテスト								
GaMe-5.1.1	K1	ゲームレベルのコンポーネントを想起する。	x					
GaMe-5.1.2	K2	ゲームレベルに典型的な欠陥を分類する。		x				
GaMe-5.2.1	K2	ゲームレベル作成のさまざまな段階で実施されたテストを要約する。				x		x
GaMe-5.2.2	K2	ゲームレベルのテストに参加する専門家の責任範囲を比較する。						x
GaMe-5.3.1	K2	ゲームレベルをテストするためのツールの使用法を要約する。					x	
6.ゲームコントローラーのテスト								
GaMe-6.1.1	K2	典型的な入力デバイスと特殊な入力デバイスを分類する。	x					
GaMe-6.1.2	K2	さまざまな入力デバイスの例を、そのアプリケーションの観点から挙げる。		x				
GaMe-6.1.3	K1	さまざまなタイプのゲームコントローラーを想起する。	x					
GaMe-6.1.4	K2	ゲームコントローラーの仕様に関連したゲーム製品の欠陥とその発生原因を分類する。		x				
GaMe-6.2.1	K2	ゲームコントローラーをテストする際のテスト条件の例を挙げる。		x				
GaMe-6.2.2	K2	ゲームのテストにおいて、UX スペシャリスト、テスト担当者、ゲームデザイナーのタスクを分類する。						x
GaMe-6.3.1	K2	ゲームコントローラーの動作をテストするためのツールの使用法をまとめる。					x	
7.ローカライゼーションテスト								
GaMe-7.1.1	K1	国際化とローカライゼーションの主な目的を認識する。	x					
GaMe-7.1.2	K1	ローカライゼーションテストのステップを想起する。			x			
GaMe-7.1.3	K2	国際化とローカライゼーションの機能を比較する。		x				
GaMe-7.2.1	K2	ローカライゼーションの欠陥とその原因を分類する。		x				
GaMe-7.3.1	K1	完全および部分的なローカライゼーションテストを認識する。				x		
GaMe-7.3.2	K3	ローカライゼーションテストの種類を分類する。						x
GaMe-7.3.3	K2	ライター、編集者、翻訳者、ローカライゼーションのテスト担当者の仕事を要約する。						x
GaMe-7.4.1	K2	ゲームのローカライゼーションテストのためのツールの使用法を要約する。					x	

11. 付録 C - リリースノート

これは、認定テスターゲームテストのシラバスの最初のリリースである。このモジュールの開発は、テストが高度に技術的なフィールドであるにもかかわらず、ゲームテストの標準が存在しなかった、インタラクティブ・エンターテイメント・ソフトウェア市場の成長に対する反応として、2020 年に開始された。

認定テスト担当者ゲームテスト開発チームは、400.000 人以上のゲーム開発フィールドの専門家の数を評価する。

認定テスト担当者ゲームテストモジュールは、同僚、雇用主、顧客の間で認知されるよう、国際的に認知された認定を取得する機会を提供する。

12. 付録 D - ゲームテスト特有の用語とその他の用語

用語名	定義
3D モデル	視覚的な、数学的形式を用いた物体の 3 次元的な形状データ。
加速度センサー	ビデオゲームコントローラーがある瞬間に空間内でどのように位置しているかをソフトウェアが判断し、情報を送信できるようにするセンサー。
アンビエント	ビデオゲームの特定の場所、状況、ステージに伴う環境音のこと。
アニメーション	静止画を高い頻度で入れ替えながら連続させることで、動いている映像であると錯覚させる技術。
バックグラウンド・サウンド	ゲームの雰囲気を盛り上げるゲーム内 BGM。
バイノーラル効果	近しい 2 つの周波数の振動の刺激を両耳に同時に与えると際に見られる聴覚の錯覚。
ボス	ビデオゲームにおいて、通常の敵よりもはるかに倒すのが難しいコンピューター制御の相手。
クライアントメカニクス	ビデオゲームのクライアント側で動作するビデオゲームのメカニクス。ビデオゲームのメカニクスも参照。
コリジョン	ビデオゲームのオブジェクトの交点を識別する技術。
カラーコーディング	識別の手段として、対象物に異なる色を付ける処理。
コンピュータ・ジェネレーテッド・イメージ（CGI）	特殊効果を生み出すための 3D コンピュータグラフィックスのアプリケーション。
コンセプト段階	中核となるコンセプトを作成し、将来のビデオゲームを説明する初期設計文書を作成することに重点を置くゲーム開発プロジェクトの初期計画段階。
コアメカニクス	プレイヤーが意図するゲーム体験を定義するビデオゲームメカニクス。ビデオゲームメカニクスも参照。
文化適応性	ある文化の特性を作り出すために、ビデオゲームの環境条件を適合させるプロセス。
ディストーション	処理中の音波の形の変化・歪み。「The Oxford Companion to the English Language」より。
格闘	ビデオゲームのジャンルの 1 つ。限られた空間と時間の中で、少数のキャラクターによる白兵戦をシミュレートする。
ファーストパーソン・シューター	銃やその他の武器を使った一人称視点での戦闘を中心としたシューティングゲームのサブジャンルの 1 つ。プレイヤーは主人公の目を通してアクションを体験する。
フレームレート	単位時間あたりに変更されたフレームの数。
ゲームパッド	ビデオゲーム機で使用される両手用ゲームコントローラーの一種。ビデオゲームコントローラーも参照。
ゲームプレイ	ビデオゲームの世界とプレイヤーとの相互作用を表現するルール。
ゲームキャラクター	ゲームの中で行動する人またはその他のもの。

用語名	定義
グレーボックス	ビデオゲームのレベル開発において、ゲームプロセスをテストするために、単色の未定義フォームから"ドラフト"の 3D レイアウトを作成する段階。
ジャイロスコープ	静止フレームにある物体の方位と角速度を測定するセンサー。
ハッキング	ビデオゲームで有利になる不正な方法。
ヘルプ	プログラムの機能を説明し、ユーザーがその機能を理解するのに役立つ文書。
歴史的正確性	ゲーム内コンテンツを現実世界のコンテンツと一致するようになぞらえること。
当たり判定。同義語：影響を受ける領域	ビデオゲーム内でのオブジェクトの衝突検出と処理のための簡略化された 3D オブジェクトモデル。
侵入不可能領域	ビデオゲームの世界で、プレイヤーが誤って利用できるようになった場所のこと。
ラグ	低速なインターネット接続、メモリリーク、CPU/RAM/HDD の過酷な利用によるビデオゲームのパフォーマンス遅延。
レベルデザイン	ビデオゲームのレベルやミッションを作成するビデオゲーム開発の一段階。
レベルエディタ	ビデオゲーム内の配置や環境を作成・編集するためのソフトウェア。
レベルオブディティール（LoD）	物体の簡略化されたコピーを作成し、異なる距離から見た物体を同じように表示する技術。
レベルプロトタイプ	概念実証として作成されるビデオゲームレベルの初期サンプル。ビデオゲームレベルも参照。
ロケール	ビデオゲームのユーザインターフェースの言語、地域、特殊な異なる環境設定を定義するオプションのセット。
ルートボックス	試合終了、経験値の獲得、またはその他のゲーム内での実績に対する仮想オブジェクトの形で授与される報酬。「Wikipedia」より
マッピング	ビデオゲーム開発において、ビデオゲームのレベルを作成するための分野。ビデオゲームのレベルも参照。
メタメカニクス	プレイヤーがビデオゲームをどのように使用するかを定義するビデオゲームの仕組み。ゲームメカニクスも参照。
移動体オブジェクト (Mob)	ビデオゲームの特定のエリアを徘徊する敵のこと。
モーフィング	ある物体が別の物体にシームレスに変化するような印象を与える視覚効果。
マルチプラットフォーム	異なるプラットフォームでソフトウェアを機能させる能力。
物語	ビデオゲームの仕組みを通して伝えられる物語。ビデオゲームのデザインも参照。
ノンプレイヤーキャラクター（NPC）	コンピューター制御のキャラクター。モブも参照。
オクルージョン	音が「閉じた」環境でどのように知覚されるかを模倣するプロセス。
プラットフォーマー	ビデオゲームのジャンルの 1 つ。足場の上でジャンプしたり、階段を上ったり、敵を倒したり、レベルをクリアするために必要なアイテムを集めたりする。

用語名	定義
プレイヤーキャラクター	ビデオゲームに登場するキャラクターで、プレイヤーが操作する。ゲームキャラクターも参照。
プレイヤー対プレイヤー（PvP）	プレイヤーが 1 人または複数のプレイヤーと対戦するビデオゲームモード。
ポリゴン	3D コンピュータグラフィックスやボリュームモデリングにおいて、多面体オブジェクトの形状を定義する頂点、辺、面の集合。
ポストプロダクション段階	ビデオゲームのメンテナンス、配信、リリース後のマーケティングの工程。
プリプロダクション段階	ビデオゲームの構成要素を企画し、文書化するプロセス。
プロダクション段階	ビデオゲームの開発プロセスの一部。
パブリッシャー	ビデオゲームの資金調達、流通、販売を行う企業。「Wikipedia」より
クエスト	ビデオゲームのジャンルの 1 つで、インタラクティブなストーリーまたはキャラクターのレベルアップのための目的ベースの活動を含む。「Wikipedia」より
レーシングホイール	ハンドル、ペダル、ギアレバーを模した両手用ビデオゲームコントローラーの一種。
ラスター	画面上で表示可能なピクセルのグリッドを表すグラフィックの一種。
リバーブ	電子的に作り出されるエコー効果。「Merriam-Webster」より
リギング	ビデオゲームにおいてモデルの骨格を作り、その後にアニメーション化させるプロセス。
ロールプレイングゲーム (RPG)	ビデオゲームの 1 ジャンルで、プレイヤーが架空の設定で登場人物の役割を演じるもの。
シーンライティング	ゲームのシーンに合わせて、キャラクターを取り囲む光の量、大きさ、色、強さなどを調整すること。
サーバーメカニクス	ビデオゲームサーバー上で動作するゲームメカニクス。ビデオゲームメカニクスも参照。
セッティング	動作が行われるビデオゲームの環境。
スキニング	ビデオゲーム内で 3D キャラクターをモデルのスケルトンに関連付けるプロセス。リギングも参照。
スキッピング	サウンドトラックの一部がスキップされたように聞こえる、バックグラウンドオーディオの不具合。
音の不連続性	予期せぬ長時間の音の欠如、または音の急激な遮断。
効果音	ビデオゲームのオブジェクトのために人工的に作られた音声や音楽以外の音。
サウンドゾーン	現実のサウンド特性を持つゲーム内エリアの作成を可能にする技術。
スピードランナー	ビデオゲームをできるだけ速くクリアしようとする試み。「Wikipedia」より
スティック	可動性の自由度が 2 つに制限されたビデオゲームコントローラー。ビデオゲームコントローラーも参照。
ストア	ビデオゲームやゲームコンテンツをデジタル配信するサービス。

用語名	定義
構造的ジオメトリ	ビデオゲームのキャラクターが動くための地形の起伏や地表面を提供する技術。
地形	ビデオゲームのキャラクターが動作するプレイ地表面に関連する構成要素の集まり。
テクスチャー	ポリゴンモデルの表面にラスター画像を重ね合わせ、色をつけたり、質感を表したりしたもの。
タイル	ビデオゲームにおいて、レベルやその他の大きなイメージを構築する元となる小さなオブジェクトまたは断片。
タッチスクリーン	画面への物理的なタッチに依存するビデオゲームコントローラー。
トラックボール	自由に回転するボールをビデオゲームの入力装置として使用するゲームコントローラー。ビデオゲームコントローラーも参照。
トリガー	イベントを開始する 1 つまたは複数のアクション。
ビデオゲーム	プレイヤーがビデオ画面上の画像を操作する電子ゲーム。「Merriam-Webster」より
ビデオゲーム機	ビデオゲーム用に設計された電子機器。
ビデオゲームコントローラー	ビデオゲームに入力するための装置。
ビデオゲームデザイン	開発中のビデオゲームのゲームプレイの構造とコンテンツを作成するプロセス。
ビデオゲームデザインドキュメント	開発中のビデオゲームの詳細な説明を含む文書。
ビデオゲームデザイナー	ゲームプレイのルールや内容を開発する責任者。ビデオゲームデザインも参照。
ビデオゲームエンジン	ビデオゲームの一部またはすべてのコンポーネントを構築するためのソフトウェア。
ビデオゲーム環境	ゲームの内容：ゲームプレイ、ゲームレベル、ゲームオブジェクト、場合によってはゲームストーリー。
ビデオゲームレベル	ビデオゲームの仮想世界内の独立した場所。
ビデオゲームメカニクス	あるルールによって定義されたビデオゲーム内のアクション。
ビデオゲームオブジェクト	ビデオゲームの中でプレイヤーが接するあらゆるオブジェクト。
ビデオゲームリソース	プレイヤーがビデオゲームの目標を達成するために重要なビデオゲームの本質または特性。
ビデオゲームステート	特定の時点におけるビデオゲーム内のすべてのオブジェクトを記述するすべてのパラメータと変数の値。
視覚効果（VFX）	現実には、物理的に存在しないスクリーン上のイメージの創作物や操作のこと。
ボリューム	音の大きさや強さ。「Merriam-Webster」より。
ホワイトボックス	ビデオゲームのレベル開発において、レベルのジオメトリをテストするために仮のモデルから 3D レイアウトを作成する段階。

13. 付録 E - 索引