

Test Plan for the Graduation Game “Flowrunners”

DADIU 2019 Team 2 (Seaside Resort), Graduation Game Project, December 4th 2019 (final revision)
QA & UR Manager Mikkel Mathiasen

Overview

This document is intended as an overview of the quality assurance and user research for the graduation game project of DADIU 2019 team 2. The plan covers the scope of testing, an overview of terminology, and the systems for evaluating and reporting found issues. The main focus of the plan is quality assurance; user research is described in general, but details will be covered in separate protocols.

Quality assurance focuses on the game’s usability (or *playability*). User research focuses on comparisons with other games and on players’ experience and *appreciation*; UR and UX.

This document is useful for the QA & UR manager for structuring the testing methodology and process, and useful for the remainder of team 2 for understanding how testing will be conducted and reported on.

Most aspects of this document are negotiable, and suggestions are welcome.

TL;DR

Gameplay and functioning of the game’s atmosphere (e.g. textures, audio, lighting) will be tested. Testing methods are primarily exploratory test charters and freestyle testing, with a few test cases for checking core functionality.

The state of the game determines which aspects are the focus of testing. Aspects of the game are classified and prioritized in four levels; critical, severe, moderate & mild.

Found bugs will be prioritized based on their severity; also in four levels.

Most QA testing will be conducted by the QA & UR manager, as well as QA & UR managers from other DADIU teams.

The methods for user testing are described generally; each case will be designed and documented individually.

Templates for test cases & charters, as well as bug reporting templates, are at the last pages.

Contents

Overview	1
Bounds	2
Definitions	2
Quality Risks	3
Bug Classification	3
Criteria & Transitions	4
Proposed Schedule & Milestones	4
Test Development & Framework	6
Benchmarking	8
UX testing	8
Templates	10
Change History	11
Appendix & Miscellaneous	12

Bounds

Main issues of the testing are gameplay and functioning of the game's atmosphere (e.g. textures, audio, lighting), both auditory and visual. A full map of the QA testing scope is on the team 2 Google Drive (Team 2\Graduation Game\QA & UR\Graduation Game Testing Scope.svg) and in the *Miscellaneous* chapter.

The design and aesthetic aspects of the game will not be tested and checked as part of quality assurance, but rather proper functioning of aspects on a technical level. On the other hand, user experience testing will likely include evaluation of aesthetics and game design, with little testing of the game's functionality.

Scope

Subjects under *quality control* are:

Included	Excluded
– Character movement – Running, jumping, sliding, climbing, etc. – Touch controls – Animation – Motion matching – Transitioning and loading levels/environments – Saving & Loading progress – Localization options for Danish & English – Interface for adjusting sound levels – Visuals and audio functionality – Assets and levels loading properly – Soundtrack & SFX playing – Win and game-over conditions – Performance on Samsung Galaxy S8	– Networking options – Localization options for languages <i>other than</i> Danish & English – Performance on systems other than Galaxy S8 – Design of visuals, audio & gameplay – Aesthetics – Level & puzzle design –

Definitions

Terms and definitions for testing the game are described here.

Test plan: this document. Contains the overall plan and strategy for the testing the project.

Test case: an individual test scenario and its script (a sequence of steps). Test cases provide an operational view of individual tests. A test case is essentially a very specific procedure for *checking* functionality.

Test archive: the complete collection of test cases.

Test cycle: cycles of tests, performed in parallel with the development cycles – i.e. a new cycle for each new major build. Daily builds means daily test cycles.

Test suite: a smaller, tailored collection of test cases, for the current test cycle – i.e. a new suite for each new cycle. See *Test Development & Framework* for a more detail on how cases, suites and the archive interact.

Test charter: a loosely scripted mission brief for exploratory testing. Test charters allow the tester more freedom in testing, and are intended to provide a framework for generating new insights and suggestions that test cases cannot, in addition to checking functionality. Test charters are time boxed (loosely).

Risk assessment/analysis: a system for assessing the risks and issues with the game's development. Features will be classified and prioritized based on a risk level, and features of a higher level will be under greater scrutiny and thorough testing than features of lower levels. See *Quality Risks* for more detail.

Bug: The working definition of a *bug* is that it is something that *bugs* someone, i.e. a bug threatens the value which the game provides to players, artists, programmers etc.; bugs are contextual. QA focuses on functionality bugs, UX & UR focuses on experience and playability "bugs".

Quality Risks

The testing scope is prioritized into four *risk levels*. These levels indicate the risk associated with not testing a particular feature (or category of features), i.e. the risk levels indicate which features are **essential** and which are **not**, from a testing perspective.

Quality Risk Assessment:

1 – Critical	2 – Severe	3 – Moderate	4 – Mild
– Character movement – Touch controls – Levels & environment	– Animation – Saving and loading progress – Loss & win conditions	– Performance – Visuals and audio functionality	– Localization – Audio configuration

Test cases will be assigned a *risk* for the associated features, i.e. its level of importance, to indicate how important it is for the test to be conducted. Critical features will need to be tested more often and thoroughly. Note that risk levels and bug severity are not the same. A bug affecting performance may be critical despite performance only having a moderate risk level. Risk levels indicate prioritization in *planning* testing, whereas bug severity indicates prioritization in *resolving* issues.

Bug Classification

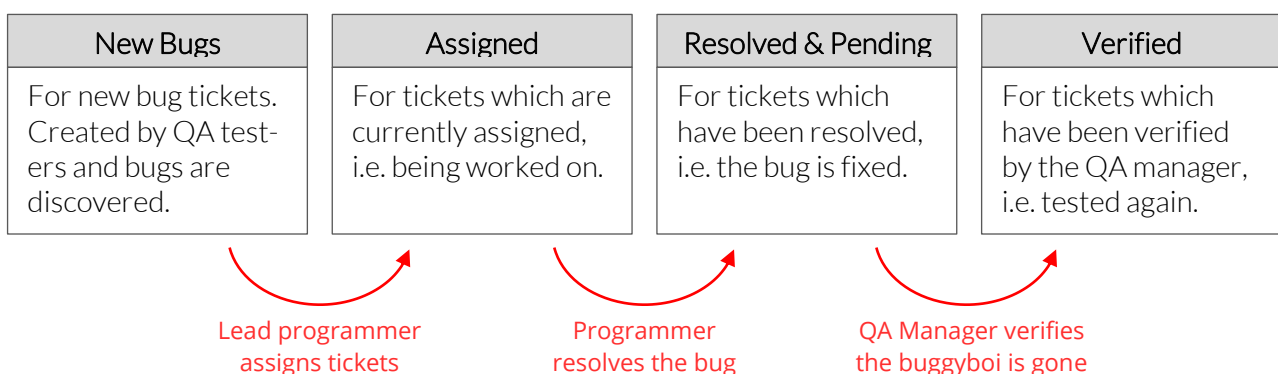
Bug classification is done with a scale for indicating the severity of individual bugs, and their priority (1 = highest priority, 4 = lowest priority). The bug prioritization has four levels as shown below. The fifth level – suggestions – is not for issues, but suggestions on aspects outside the scope of the test plan.

Bug Severity Classifications:

1	Critical bug	Gamebreaking. You can't play the game
2	Severe bug	Not game breaking but severely impacts the game
3	Moderate bug	Impacts gameplay
4	Minor bug	Small, does not impact the gameplay
5	Suggestion	Not a bug

Bug Tracking Tool

Bugs will be tracked on a board on Trello.com (<https://trello.com/b/LdGEemz1/team-2-buggyboard-ggp>), where each found bug is assigned to its separate card, and moved between four different columns, depending on the stage of resolving the issue. There are four stages to solving a bug: finding & documenting the bug --> assigning it as a programming task --> actually fixing the issue --> and finally verifying that the issue has been resolved in a new build.



Criteria & Transitions

QA testing should be done on a working prototype at least in a state of alpha; i.e. with core mechanics, but otherwise no finished assets and levels are needed.

Having different phases for testing is intended to focus testing on version-appropriate aspects of the game, e.g. basic performance, controls and mechanics during alpha, expanded to assets, levels and animation during beta. The criteria for entering and continuing these phases are intended to ensure that testing can be done effectively and continuously.

Entry Criteria describe the conditions to be met for a particular phase to begin.

Continuation Criteria describe the criteria to be met to continue testing during that particular phase.

Exit Criteria describe the conditions to be met to transition into the next phase (essentially just the entry criteria for the proceeding phase).

A unique **identifier** for each build is important for tracking bugs. The identifier could be a series of numbers or it could be a unique generated name.

The **changelog** should contain changes from the previous game version in some form.

The below criteria for testing different versions of the game are *suggestions*.

Proposed main phases and their criteria for GGP:

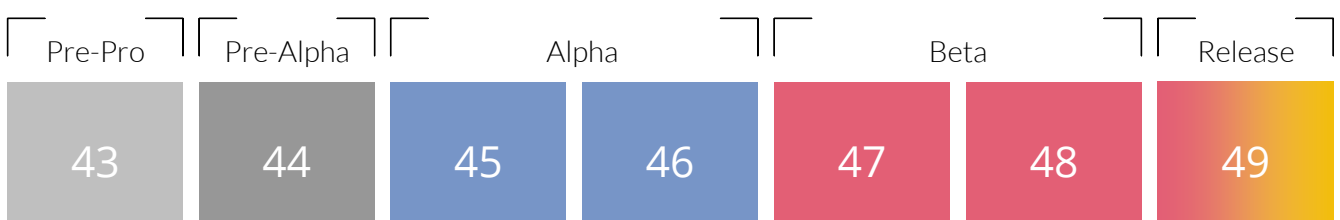
	Alpha	Beta	Gold
Entry	– Working prototype with some core functionality – No finished assets, textures or sounds needed (any placeholder acceptable)	– Core functionality in – All assets in (not necessarily finished) – No critical bugs from Alpha	– Finished game – All assets and functionality finished and implemented – No remaining (detected) critical and severe bugs – Minimum number of major and minor bugs
Continuation	– New builds Must: – Have a unique identifier – New builds <i>should</i> – Include a changelog	– New builds Must: – Have a unique identifier – Have debugging tools – New builds <i>should</i> – Include a changelog	
Exit	– Beta entry criteria	– Gold criteria	

Gold criteria, here, are also the general *testing exit criteria*, i.e. the criteria for the game to be considered *done*. At this point, no more QA testing is necessary, but may still be conducted for polish and evaluation.

Proposed Schedule & Milestones

Suggestions for project milestones related to testing. Refer to the production plan for a more comprehensive overview. This section exists both for planning purposes and as a retrospective. It is to be updated with further user testing and when/if milestones are moved.

A rough outline of the production looks something like this (boxes represent weeks);



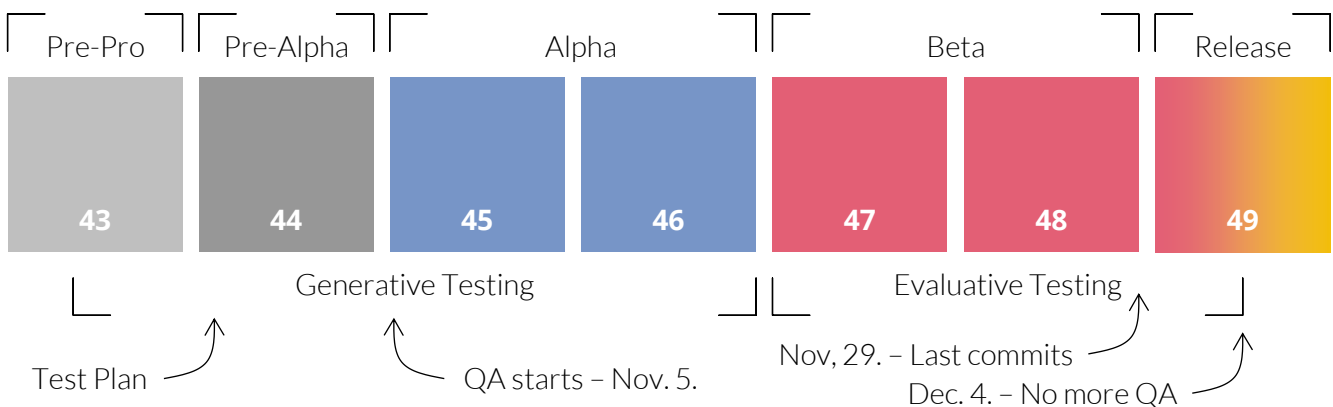
Milestones and UX/UR Tests

Proposed dates for transitions and tests.

- **October 18th** – ZMET wit single participant
- **October 25th** – Initial benchmarking done – Runner games general
- **October 28th** – Production starts
- **October 29th** – Test plan finalized
- **November 4th**
 - Game enters alpha
 - QA testing starts (latest)
- **November 4th – 7th** – Generative testing on game feel and the concept of ‘parkour’
 - Protocol: Team 2\Graduation Game\QA & UR\ Generative Session Protocol 02.pdf
 - Three Generative sessions – Four participants
- **November 5th** – QA testing starts
- **November 12th** – Generative playtesting on alpha build
 - Protocol: Team 2\Graduation Game\QA & UR\ Playtest Protocol 02.pdf
 - Single playtesting session – single participant
- **November 19th** – Game enters beta
- **November 19th – 21st** – Playtesting on beta build
 - Protocol: Team 2\Graduation Game\QA & UR\ Playtest Protocol 03.pdf
 - Three playtesting sessions – three participants
- **November 25th – 27th** – Playtesting on beta build
 - Protocol: Team 2\Graduation Game\QA & UR\ Playtest Protocol 04.pdf
 - Four playtesting sessions – six participants
- **November 29th** – last commit – no more art and game design added from this point
 - Work starts on a release candidate
- **December 4th** – QA Testing ends
- **December 6th** – Final reports on the testing process
 - Quality report and user acceptance report – two *separate* documents
 - Including the final revision of this test plan

Simplified Timeline

A simplified view of the testing process over time looks like this;



Test Development & Framework

Testing will be conducted primarily by the QA & UR manager, and Trello.com will be the primary tool for organizing and reporting upon test cases and suites.

Key Planning Personnel: QA & UR Manager, programmers, directors and project manager. Planning tests cycles and user tests.

Key QA Testers: QA & UR Managers – QA & UR managers will regularly act as QA testers for other teams, going through a test suite and build provided by another QA manager. Thus team two will provide QA testing for other teams, and have tests performed by other teams in return.

Test hours: QA testing between 9:00 and 17:00, whenever appropriate, UR and UX testing whenever participants are available.

Daily builds: in the case of daily builds, testing *should* be conducted every day, however daily testing should be limited to newly added and critical functionality.

Test log: all testing should be logged in a separate document; both UX & QA testing. The log should provide details on each test (date, participants, cases, etc.), and for QA testing the number of found issues should be logged for evaluation purposes.

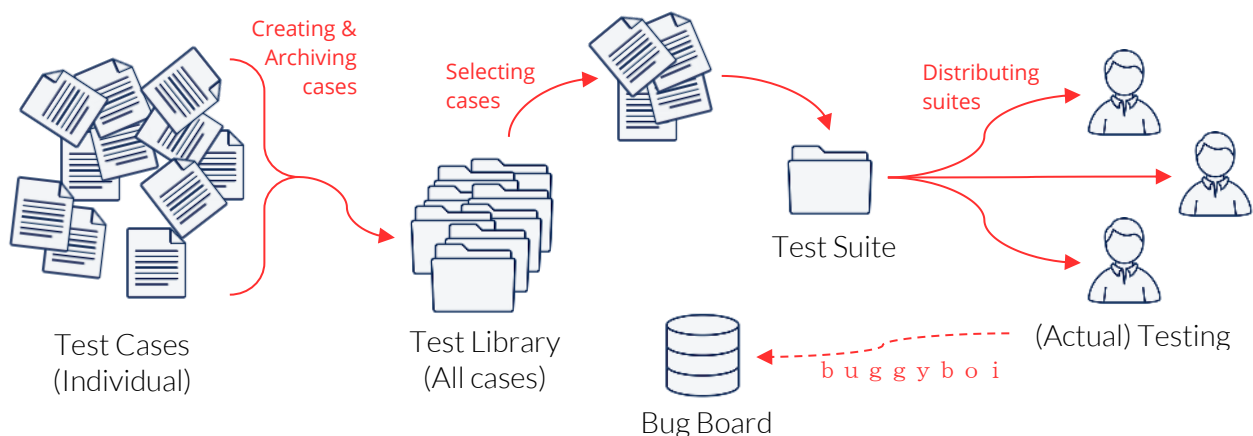
Primary QA testing methods

The primary methods for QA testing will be test cases and test charters. Performing test *cases* will check particular functionality in a particular way. Cases are simple and easily repeatable, making them useful for testing core mechanics and interfaces. Performing and iterating upon test *charters* will explore the performance and content of the game in a contextual way; much less directed as cases, but much more sensitive to changes and unforeseen issues. Test cases should be performed early and quickly in each test cycle, followed by more extensive testing with charters.

The particular frameworks for cases and charters are described below.

Anatomy of the testing framework

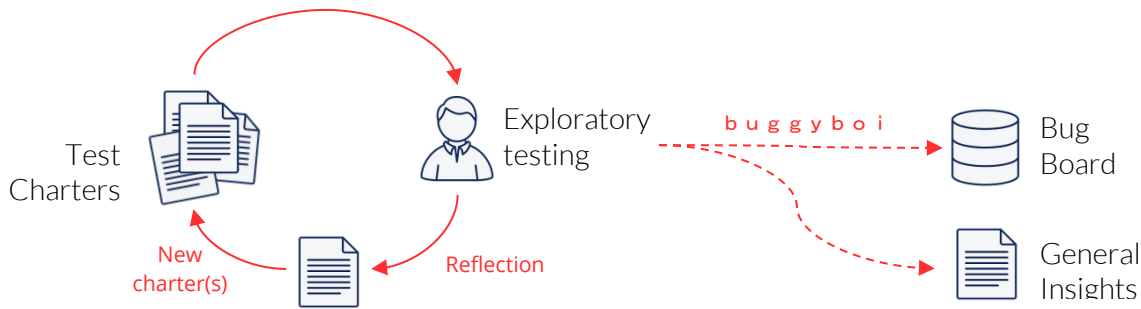
The “pipeline” for test cases and their implementation looks something like this (though there will often only be a single tester) ↓



Test *cases* and test *library* persists across test cycles (though they may be expanded), while a new selection of cases, i.e. a test *suite*, is compiled each cycle, and then performed.

Charters

The framework for test charters is iterative, as new charters are produced in response to performing existing charters ↓



Test library and suites

The test library is managed through a board on Trello.com, with cases represented by individual cards. The test library *should* only be accessible to QA & UR managers, but for the sake of convenience the library will be kept on the same board as bugs are being reported on, in the rightmost column.

Test suites for each test cycle are to be put together by the QA & UR Manager based on the changelog for new builds as well as the quality risks. Test suites will be performed by the QA & UR manager each cycle, as well as other testers when available.

Test cycles

Each test cycle goes through a process like this:

1. New build
2. Test suites put together by QA manager based on changelog and quality risks
3. Suites performed by QA manager and potentially others
4. Testing continues with test charters of whatever is appropriate
5. New test cases may be created on the basis of prior testing and changes
6. Any user tests takes place when planned
7. Freestyle testing and/or research until next build
 - E.g. freestyle exploratory testing, competitive analysis, observational user research, etc.

Bug Reporting

Only severe and critical bugs will be reported before and during alpha. During beta all types of bugs will be reported.

Bug tracking will be done through a bug ticket board on Trello.com. Critical bugs will be reported immediately to the lead programmer, as these may prevent further testing.

Regression testing

Regression testing will be done on core functionality through test cases and charters. Regression testing will be conducted more often on functionality with *critical* or *severe* risks (see *Quality Risks*).

Test Environment

Quality assurance tests will be conducted in the team 2 production area, or wherever appropriate. Any potential user tests will be conducted wherever appropriate. Tests will be conducted in the provided hardware platform; Samsung Galaxy S8 running Android. For user tests for later projects, some contextual and mobile methods may require settings such as parks and other public areas.

Benchmarking

Benchmarking lies in between QA and UR; benchmarking is a process of mapping and analyzing other games on the market which informs both UR and the scope of QA, and at the same time benchmarking investigates mechanics and levels of specific games, which can directly inform game & level design. Finally, a benchmark of the final game can help understand its qualities in relation to existing games; whether it turned out as intended or not.

- Benchmarking is primarily intended to inform game design and direction
 - Mapping the market of games related to the core concept
 - Analyzing similar games' mechanics and levels
- Aspects of similar games can inform QA testing
 - When the project is in its early stages, other games may provide a reference for scoping tests
- Aspects may also inform UX testing and form the basis for comparisons

For documents related to benchmarking, see the Google Drive folder:

- Team 2\Graduation Game\QA & UR\Benchmarking

UX testing

This chapter covers the UR & UX aspect of QA & UR manager's responsibility. Details will be light, as each session will be planned and documented separately (if at all), but the purpose and methods of user testing are important to establish.

UX testing is, unlike QA testing, not for finding bugs, but for finding killjoys and generating feedback and ideas about mechanics, controls, graphics, difficulty, social aspects, etc. The aspects investigated through user- and playtesting will likely go beyond the scope of quality assurance as describe in *Bounds*.

UX testing can be done *with* and *without* a working build. In early stages of development tests should be conducted on concepts and sketches, and findings would be generative, i.e. participants produce maps and sketches which may inform design. In later stages of development, testing with users should be done on the basis of a working build, accompanied by some framework for expressing and capturing their opinions and reactions; these later tests would be far more evaluative than generative.

Since QA testing is focused on the functional aspects of the game, it makes sense for tests with users to focus on their experiences and the experiential aspects of the game. For this purpose generative and visuals based methods are useful; the methodology includes Generative Tools/Toolboxes (Sanders), Visually Stimulated Discussion (Weber), Metaphor Elicitation Technique (Zaltman) & Mental & Conceptual Mapping

Protocols and testing mentality

Tests with users will be conducted on the basis of a protocol made beforehand. Some protocols may be very specifically directed at a single test, and some may be used for several separate tests. A protocol should state the purpose of a given test, and contain some semblance of a script or procedure; a sequence of questions and activities.

While protocols may be very specifically planned, it is not intended to be followed meticulously; each participant will have their own opinions and a particular player profile which impact how they play and their thoughts on the experience. Performing UX tests will require agility and responsiveness from the researcher, more so than requiring that the protocol be followed with precision.

Reporting on playtesting

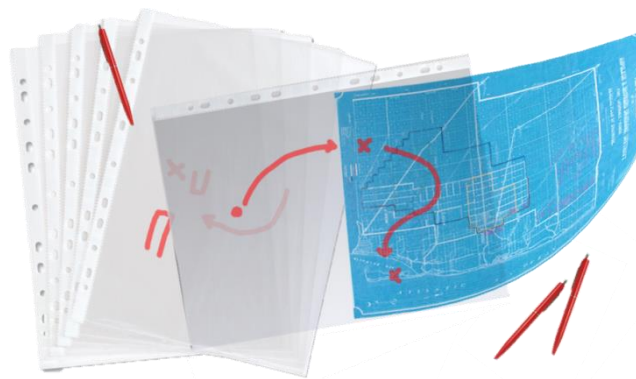
Findings from tests with users may be compiled into reports, synthesized so that they are easily and quickly read. These reports could take the form of slides, but more than likely they will be verbal.

User testing may also feed into market research and QA, e.g. terms uncovered with ZMET may form the basis of a benchmark.

Generative tools

User testing could make use of generative tools. After (or in conjunction with) playing the game, participants could be presented with a generative toolset consisting of printouts of props and levels, illustrations and drawing tools, which would then serve the participant in expressing their experience in narratives and/or visual compositions. A short, post-playtest, generative “session” may just be a single question or reflection which is visualized on a physical board or model.

A generative toolset *could* consist of level schematics and concepts, markers and plastic envelopes; participants could mark the schematics/concepts through the envelopes with concrete changes and suggestions, or with experiential notes. It could look like this ↓



The generative tools are based on these texts:

- Sanders, E. (2002). From user-centered to participatory design approaches. In J. Frascara (Ed.), *Design and the Social Sciences: Making Connections* (2005th ed., pp. 1–7). New York: Taylor & Francis.
- Sanders, E. B.-N. (2000). Generative Tools for Co-designing. In S. A. R. Scrivener, L. J. Ball, & A. Woodcock (Eds.), *Collaborative Design* (pp. 3–12). https://doi.org/10.1007/978-1-4471-0779-8_1
- Sanders, L., & Stappers, P. J. (2012). How it works: Generative tools and techniques. In *Convivial design toolbox: Generative research for the front end of design* (pp. 65–95). Amsterdam: BIS.

Visually Stimulated Discussions

Instead of generating visual material, participants may engage with the QA & UR manager over visual content from the game’s development. Concept artwork, screenshots, videos of prototypes, etc. would be shown to the participant before or after playing the game. These visual materials would serve as an outset for a discussion and reflective session.

This could also be combined with a generative toolbox.

This method is based on: Weber, S. (2008). Visual Images in Research. In J. Knowles & A. Cole, *Handbook of the Arts in Qualitative Research: Perspectives, Methodologies, Examples, and Issues* (pp. 42–54).

Mental & Conceptual Mapping

The three above mentioned methods – generative tools, visually stimulated discussions and ZMET – will be conducted with the purpose of forming a visual and conceptual map of players understanding of and associations with particular concepts. The map will likely be the primary product of these tests, alongside the generated materials.

ZMET

Zaltman's Metaphor Elicitation Technique (ZMET) is a type of generative tool based on images, concepts and *constructs*. This method asks participants to bring ten pictures relating to a specific concept, followed by a process of sorting, reflecting and combining the participants' images. The method is intended to *elicit metaphors* and associations through the process of narrating and organizing the collection of images. There are ten steps ↓

1. Storytelling: describe the content of each picture
2. Missed images: describe pictures that was impossible to obtain and explain relevance
3. Sorting: sort pictures in meaningful piles and label them
4. Constructs: use pictures to derive constructs (Kelly Repertory Grid and Laddering)
5. Most representative picture and "widening" (what other elements could be part of the picture)
6. Opposite images: user describes pictures that are the opposite of the topic
7. Sensory images: describe tastes, smells, touch, sounds, colours and emotions attached to the topic
8. Mental map & vignette: bring the participant's unconscious thoughts to a level of awareness
9. Collage: a summary image or montage expressing the topic under investigation
10. Consensus map: highlights common themes and threads showing relationships among constructs*

Appreciation Testing

An appreciation test (or *user acceptance test*) is intended to gauge the players' overall impression of the final game; its playability, learnability, memorability and players' enjoyment. This will be the latest form of test and should be conducted with a version of the game as close to gold as possible. Unlike previous tests, appreciation test will only be evaluative.

Selecting Participants

Users will not be selected thoroughly based on demographics or a questionnaire, instead, participants will fill out a psychographic survey, i.e. Quantic Foundry's *Gamer Motivation Profile* survey, so that findings can be compared on the basis of their player profile.

Templates

Templates for test cases, bug tickets, the bug board and test charters,

Test Case Template

The test cases will look similar to the table below, and contain the same points, but will be organized on Trello.com.

Team nr.	Team 2	
Feature	(to test)	
Summary	Very short	
Risk (priority)	1-4 (1-5) – testing priority essentially	
Allotted time	X min.	
Environment		
Test type	Unit, component, integration, system or test	
Description	The function and scenario – <i>specific</i>	
Expected result(s)	What <i>should</i> happen if the function works	
Script	1	Do this..
	2	Then that...
	3	...
	4	...

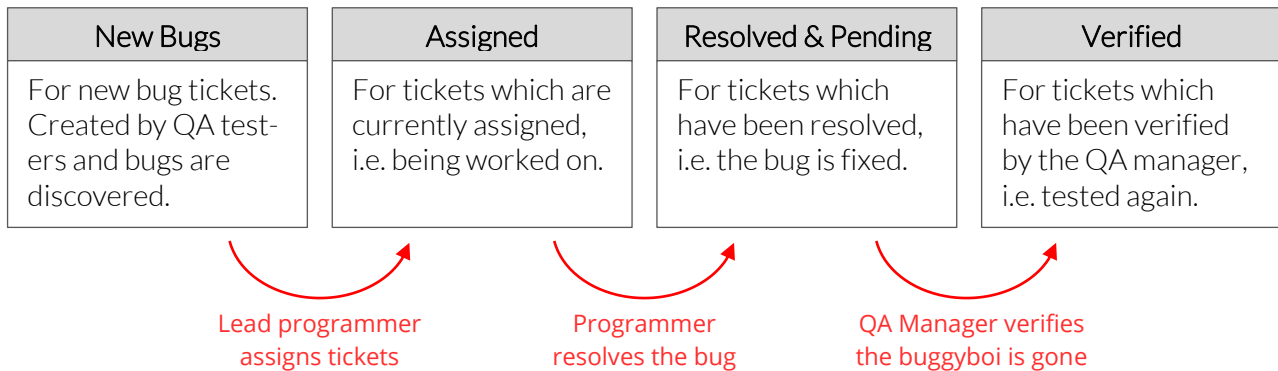
Bug Ticket & Board Template

The bug tickets will look similar to the table below, and contain the same points, but will be organized on Trello.com

ID	build;bug or case;build;bug (e.g. 0.12.4;07)	
Priority/risk	1-5	
Description/ summary	What went wrong	
Expected result	What <i>should</i> have happened	
Frequency	X in Y instances or X%	
Steps to reproduce	1	Do this...
	2	Then that...
	3	...
	4	...
Notes	Suggestions, workarounds etc.	
Attachments	Linked images or videos illustrating the issue	

Bug board

The bug board looks and works like this:



Test Charter Template

A test charter looks something like this:

Mission	The mission statement – feature
Scope	Short summary of features to explore
Allotted time	X min.
Risk (priority)	1-4
Notes	For notes <i>during</i> and <i>after</i> the test session

Change History

Here goes changes to the test plan; newest changes first.

November 25th to December 4rd

- Updated Overview
- Updated *Bug Classification* - now its own chapter
 - Added bug tracking to this chapter
- Updated *Proposed Schedule & Milestones*
 - Now all tests are in the schedule – retrospectively
- Updated section on *Test library and suites*

- Added *Protocols and testing mentality* to *UX Testing* chapter
- Updated the test case template
- All templates are now in a single chapter
- Added screenshots from Trello.com to the appendix
- Moved around sections for readability
- General formatting improvements

November 4th to November 25th

- Updated schedule – milestones and timeline
 - Added testing sessions retrospectively

October 28th to November 4th

- Updated schedule

September 23rd to October 28th – Changes from the MGP2 test plan

- New Scope to fit new concept
- New proposed milestones to fit new production plan
 - Added simplified timeline for quick overview
- Moved *Bug Classification* into *Quality Risks*
- Updated definition of “bug”
- Changed alpha and beta continuation criteria
- Updated bug ticket template
- Updated the entire *UX testing* section
 - Added *Appreciation testing*, *ZMET*, *Selecting Participants* and *Mental & Conceptual Mapping*
 - Removed *Art-based Research & Autoethnography*
- Moved several sections around for easier reading

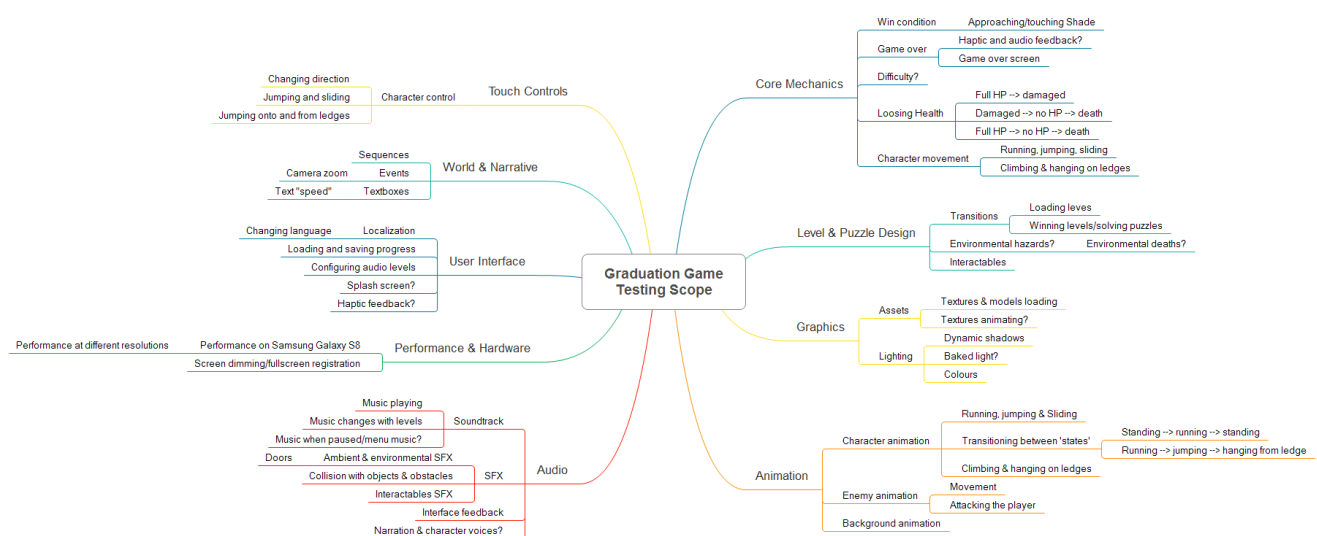
September 9th to 23rd – Changes from the MGP1 test plan

- Added test charters (along with a template)
- Added contents to the overview
- Updated scope to fit new concept (in development)
- Added another quality risk level – now bug severity and quality risks both have four levels
- Updated proposed milestones
- Expanded the test development and framework
- Added an overview of the bug board
- Added initial versions of playtesting and generative testing

Appendix & Miscellaneous

He be the QA scoping map and examples of test cases and bug tickets.

QA Testing Scope Map (October 29th)



Test cases as they appear on Trello.com (on December 4th)

Case 1;2 - Level Loading & Transitions

[in list Test Cases](#)

LABELS

Critical +

Description

Edit

Team Number: Team 2

Feature: Loading and transitioning between levels

Summary: Progress through the entire game to verify that all levels load in proper order

Risk: Critical - 1

Estimated Execution Time: 10-15 min.

Environment: *Wherever*

Description: Progress through all levels, including level 0/splash screen

Expected Result(s): All levels load in proper order with all their assets. Upon clearing one level the player should progress to the next without issue.

Script/procedure:

1. Launch the game and start a 'new game'
2. Clear level one and progress to the immediately proceeding level, i.e. level 2
3. Repeat for each level until the game has been progressed through in its entirety, i.e. all levels have been cleared

Note: Total levels are: 0 (tutorial) --> 1.1 --> 1.2 --> 2.1 --> 2.2 --> 3 --> 4

Activity

Hide Details

Case 1;3 - Basic Character Movement & Controls

[in list Test Cases](#)

LABELS

Critical +

Description

Edit

Team Number: 2

Feature: Running, jumping & sliding

Summary: Check that running, jumping and sliding works in both directions.

Risk: Critical - 1

Estimated Execution Time: 5 min.

Environment: *Wherever*

Description: Check that running works, and that the direction can be changed by swiping in the opposite direction. Check that jumping works. Check that sliding works

Expected Result(s): Running, jumping and sliding should work as intended, and controls should behave as described (refer to the *character controls* card in the first column)

Script/procedure:

1. Check the *character controls* card (leftmost column)
2. Launch the game and load any level
3. Observe as the character starts moving automatically
4. Jump (swipe upwards in the running-direction)
5. Slide (Swipe downwards in the running direction)
6. Change direction and perform nr. 4. & 5. again
7. Stop the character against a wall
8. Change direction while stopped against a wall
9. Change direction while running

Activity

Hide Details

Case 2;2 - Saving progress

[in list Test Cases](#)

LABELS

Severe +

Description

Edit

Team nr.: Team 2

Feature: Saving progress

Summary: Progress should be saved between sessions

Risk: Severe - 2

Allotted time: 2 min.

Environment: *Wherever*

Description: Launching the game should load the most recently available level; the game remembers the latest completed level

Expected: The latest, unfinished level will load

Script:

1. Launch the game
2. Progress to level 2 (or any later level)
3. Exit and relaunch the game
4. Verify progress by continuing to the latest level
5. Progress to the next level
6. Exit and relaunch the game
7. Verify progress by continuing again

Activity

Hide Details

Case 3;2 - Soundtrack

[in list Test Cases](#)

LABELS

Moderate +

Description

Edit

Team nr.: Team 2

Feature: Soundtrack

Summary: Verify that the soundtrack plays

Risk: Moderate - 3

Allotted time: 15 min.

Environment: *Wherever*

Test type: Component test

Description: Listen for background music in each level

Expected: Background music should play in every level (and change with each level)

Script:

1. Start up the game and progress to level one - listen for music
2. Progress to level two & listen for music
3. Progress through each proceeding level and listen for music

Activity

Hide Details

Case 4;1 - Localization options

[in list Test Cases](#)

LABELS

Minor +

Description

Edit

Team nr.: Team 2

Feature: Localization options

Summary: Switching between Danish and English

Risk: Mild - 4

Allotted time: 1 min

Environment: *Wherever*

Test type: Component test

Description: Change language between Danish and English. The currently selected language should be highlighted

Expected: The interface language corresponds to the selected language

Script:

1. Launch the game
2. Open the settings panel by tapping the cogwheel
3. Select the language not in use
4. Check if the interface corresponds with the change
5. Revert to the original language and check again

Activity

Hide Details

Case 4;2 - Audio configuration

[in list Test Cases](#)

LABELS

Minor +

Description

Edit

Team nr.: Team 2

Feature: Audio configuration

Summary: Adjust audio level

Risk: Mild - 4

Allotted time: ½ min.

Environment: *Wherever*

Test type: Component test

Description: Adjust audio level

Expected: The audio level change accordingly

Script:

1. Launch the game
2. Open the settings panel by tapping the cogwheel
3. Drag the slider to adjust the audio level
4. Listen for the change
5. Adjust again and listen

Activity

Hide Details

Big tickets as they appear on Trello.com (on December 4th)

dev_102;02 - Game crashing loading into lvl 1.2
in list [Verified](#)

MEMBERS + LABELS **Critical** +

Description Edit

Build and bug nr.: dev_102;02
Risk (priority): Critical - 1
Description: When loading into lvl 1.2 the game crashes. This happens both when loading from level select and when finishing lvl 1.1
Expected result: no crash plz
Frequency: 100%
Steps to reproduce:
- Launch the game
- progress to lvl 1.2 (if you can >:))
- profit?
Note: Also in dev_104

dev_89;09 - Spike & railing in lvl. 1.2 are offset
in list [Verified](#)

MEMBERS + LABELS **Minor** +

Description Edit

Build and bug nr.: newdev_89;09
Risk (priority): Mild - 4
Description: Spikes & railing on the top red platform in lvl 1.2 are offset and thus do nothing (maybe intended)
Expected result: I dunno, may be intended?
Frequency: 100%
Steps to reproduce:
- Launch the game, load lvl 1.2 and observe
Notes: May be intentional?

dev_121;03 - Big elevator in lvl 2.1 does not properly respawn
in list [Verified](#)

MEMBERS + LABELS **Severe** +

Description Edit

Build and bug nr.: dev_121;03
Risk (priority): Severe - 2
Description: If Zoe dies after triggering the last, big elevator, it becomes unable to be triggered again, i.e. it does not move again.
Expected result: The elevator should reset and be able to be triggered again
Frequency: 100%
Steps to reproduce:
- Launch the game and load level 2.1
- Progress to the end of the level, and get onto the large red elevator
- Make zoe die while on the elevator, and respawn
- Get onto the elevator again and observe

Attachments

LINK **lvl 2.1 elevator**
Added 22 Nov at 11:52 - [Comment](#) - [Remove](#) - [Edit](#)

dev_260;01 - Textures missing in some levels
in list [New buggybois](#)

MEMBERS + LABELS **Moderate** +

Description Edit

Build and bug nr.: dev_260;01
Risk (priority): Moderate - 3
Description: shit's purple
Expected result: proper textures
Frequency: 100%

Attachments

Screenshot_20191128-101435_GraduationGame.jpg
Added 2 Dec at 12:58 - [Comment](#) - [Delete](#) - [Edit](#)
[Make Cover](#)

Screenshot_20191128-100534_GraduationGame.jpg
Added 2 Dec at 12:58 - [Comment](#) - [Delete](#) - [Edit](#)
[Make Cover](#)

Screenshot_20191128-100405_GraduationGame.jpg

View of the bug board (on December 4th)

Team 2 Buggyboard GGP | Team 2 | Free | Card Colors: Merge | Public | +5 | Invite | Card Colors: Merge | Butler | Slack | Show Menu

Templates & Formatting Help | Bug Classification | Board Format | Character Controls | Identifier Format

New buggybois

- dev_260;03 - In lvl 3, Zoe is stuck after speaking with the stranger
- dev_294;01 - Missing dialogue in 2.1
- dev_260;04 - Sometimes dying makes Zoe do freaky looped "animations"
- no_debug;03 - General music and sound issues
- dev_241;02 - Elevators can push Zoe into the levels' walls
- dev_260;05 - Lasers in lvl 2.2 may not collide properly still
- dev_286;01 - Low framerate in lvl 4
- dev_151;03 - Zoe can jump onto spikes and get stuck
- dev_260;01 - Textures missing in some levels

Assigned/In Progress

- dev_89;06 - Zoe's animation do not work properly

Resolved & Pending

- + Add a card

Verified

- dev_151;04 - Zoe can get stuck atop the police-man-bean-thingy
- Dev: 89;18 First red platform is out of view
- dev_89;07 - Checkpoint in lvl 1.2 does not work
- Dev: 89; 18-11: Checkpoint in level 2 not working
- Dev: 89;18 Menu: setting button
- dev_89;09 - Spike & railing in lvl. 1.2 are offset
- dev_100;02 - Game crashing loading into lvl 1.2
- dev_89;02 - Game doesn't seem to save at all
- dev_89;01 - Zoe can ledge-grab while rolling

Test Cases

- Case 1;2 - Level Loading & Transitions
- Case 1;3 - Basic Character Movement & Controls
- Case 2;2 - Saving progress
- Case 3;2 - Soundtrack
- Case 4;1 - Localization options
- Case 4;2 - Audio configuration

hmmm