

Quality Report & Retrospective for “Flowrunners”

DADIU 2019 Team 2 (Seaside Resort), Graduation Game Project, December 6th 2019

QA & UR Manager Mikkel Mathiasen

Overview

The quality assurance testing for Flowrunners was based on a mix of process-driven and context-driven methods, as introduced in workshops with Benjamin Legarth on September 4th, 5th & 6th 2019, and with Claus Petersen on September 19th & 20th 2019.

In terms of describing the methodology, I have simply pasted in three sections from the Test Plan; *Bounds*, *Bug Classification* and *Test Development & Framework*. The sections on *Process* and *Reflections* should clarify some of the underlying thoughts behind the methods.

In short, the game was tested using mostly exploratory test charters, with a few test cases for testing specific core functionality. The scope of testing was prioritized on four risk levels, from mild to critical. Equally, found issues were assigned one of four risk levels, depending on their impact on gameplay. I kept a log of all testing, logging the number of issues found with both cases and charters. The found issues were shared with the programming team through a board on Trello.com

I have included examples of test cases and charters as they appeared in the testing library, and some screenshots of the bug tracking board.

The test plan, testing log, full size versions of models and other documents are available from my Google Drive folder here: https://drive.google.com/open?id=1ICZsVqCMyJ_Ev6s3ltvAtM7apwqNUefa.

Contents

Overview	1
Bounds (Test Plan excerpt)	1
Bug Classification (Test Plan excerpt)	2
Test Development & Framework (Test Plan excerpt)	3
Process Review	5
Case & Charter Examples	6
Bug Tickets & Bugboard	9
Test Results.....	10
References.....	12

Bounds (Test Plan excerpt)

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 –

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 (Test Plan excerpt)

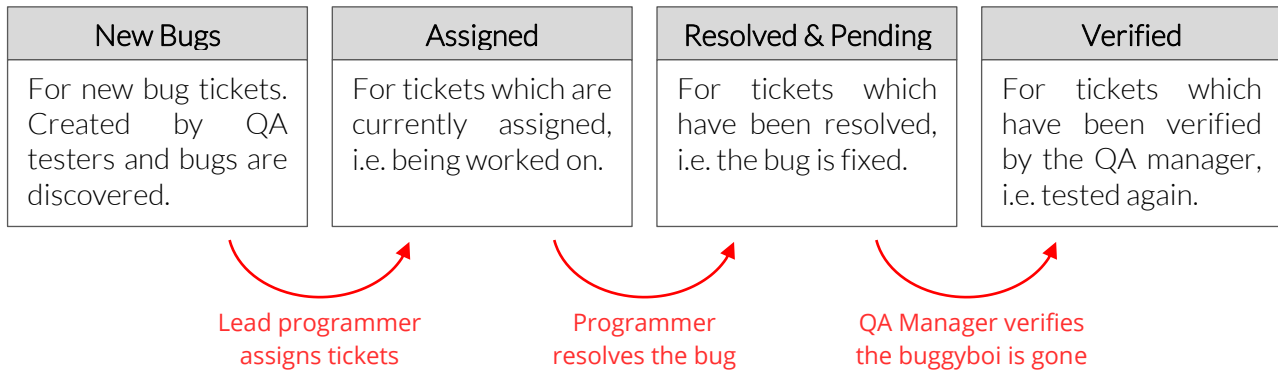
Bug classification is done with a scale for indicating the severity of individual bugs, and their priority (1 = highest priority, 4 = lowest priority). The bug prioritization has four levels as shown below. The fifth level – suggestions – is not for issues, but suggestions on aspects outside the scope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



Test Development & Framework (Test Plan excerpt)

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.

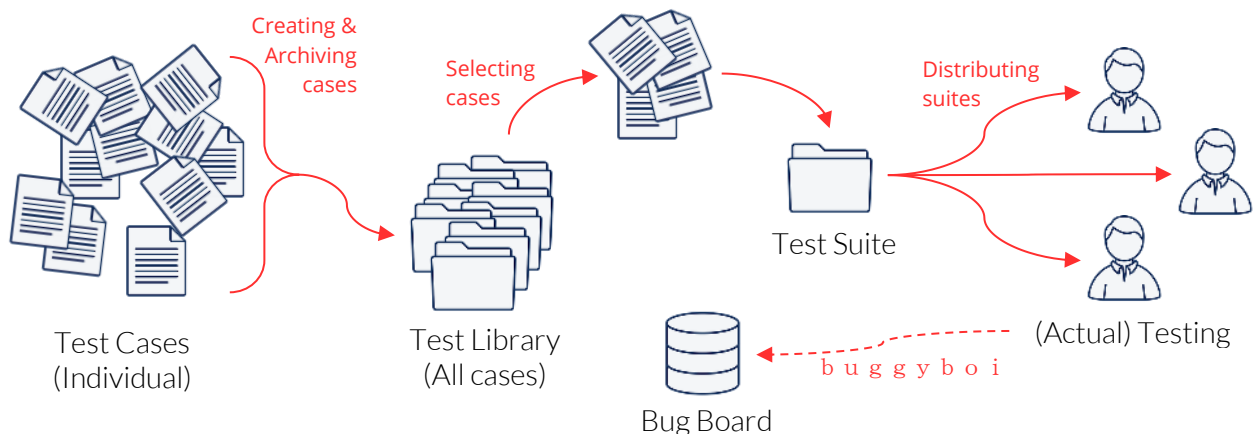
Primary QA testing methods

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

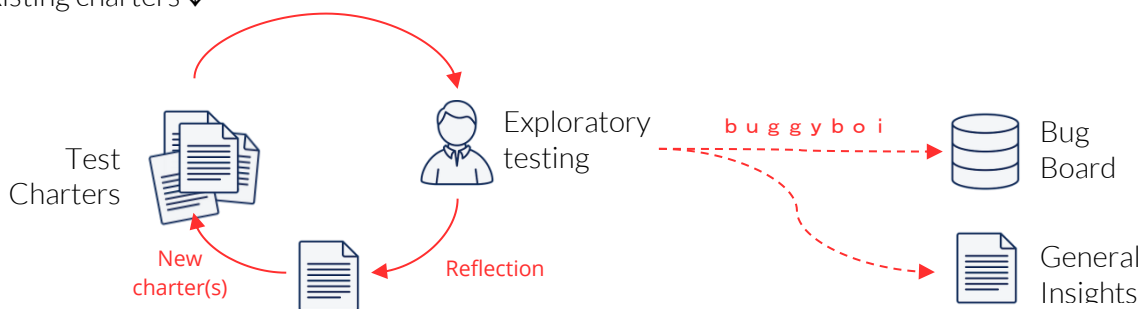
The “pipeline” for test cases and their implementation looks something like this (though there will often only be a single tester) ↓



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

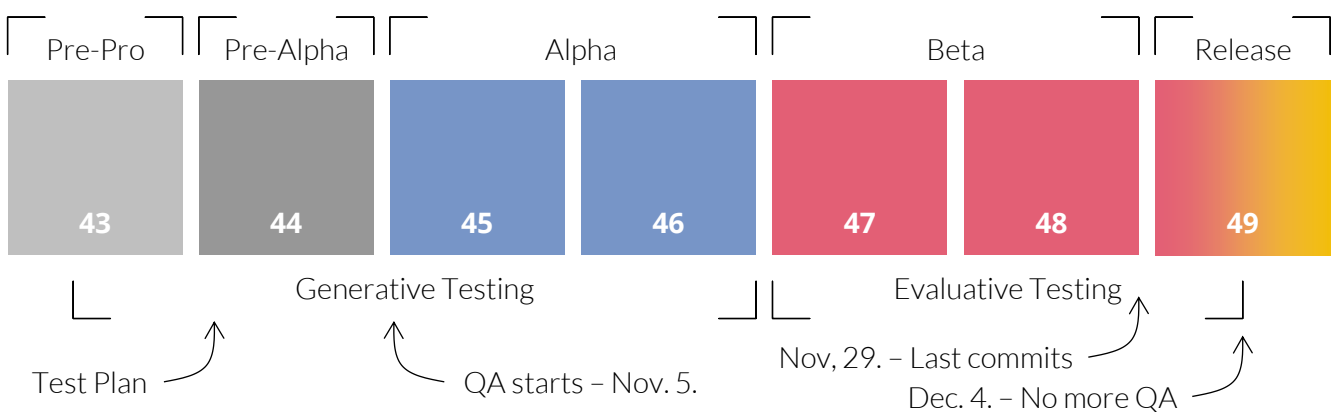
Quality assurance tests will be conducted in the team 2 production area, or wherever appropriate. Any potential user tests will be conducted wherever appropriate. Tests will be conducted in the provided hardware platform; Samsung Galaxy S8 running Android. For user tests for later projects, some contextual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

Process Review

The test plan is based on Rex Black's (2002) template, but modified greatly to accommodate UX testing. The purpose of the plan was to serve as an overview of the entire process and methodology for the team, and for myself, as QA & UR Manager, it served as a planning tool for when and how to perform tests.

With the little to no experience I had with QA going in to this project, the test plan served as an important tool for me to organize my methods and planning when to use them. I doubt that it was of much use to the team, if any at all, but at least I understood how to approach the testing process, particularly the specific methods for identifying, classifying and reporting on bugs.

Below is a visual overview of the testing process. There is not much to say about it, except the fact that I did not stop testing on December 4th, because the testing *exit criteria* (described in the test plan) had not been reached.



Methodological Considerations

I used a mix of process-driven and context-driven testing methods, the difference being the methods' replicability and agility. In our workshop with Claus Petersen, process-driven and context-driven testing were described as two separate paradigms of testing; the first – process-driven – striving for predictability and replicability of standardized tests, while the second – context-driven – values multidisciplinary, responsiveness and exploration. Test *cases* are process-driven, while test *charters* are context-driven

Petersen made another distinction between cases and charters, in that cases are a form of *checking*, and charters are better described as *testing*. This distinction is important as test cases are “tests” insofar as they are a way of directing tests at *checking* that something works as described, but they do not generate insights or explore other possible outcomes. Test *charters* on the other hand are unscripted and exploratory; the tests may generate new insights and are more sensitive to changes to the project, however they are impossible to properly replicate.

In using both test cases and charters I attempted to ensure that core functionality was tested thoroughly and repeatedly (with scripted cases), while also allowing for exploratory testing of prototype functions, level layouts, interfaces etc. (with exploratory charters)

Logging Sessions

I kept a log of all of my testing session, both QA and UR. The entire log document is in the driver folder linked in the Overview section. For each QA entry I made sure to log the specific cases and charters that I performed, and the number of issues that each of these were able to find.

An entry into the log for a QA session looks like this, with the coloured columns to the right being the number of found issues with each case/charter, separated into their severity level:

November 18 '19 - dev_89

	Test case/charter	Coverage	Issues				
1	Case 1;2 & 1;3	n/a	1	1	1	1	1
2	Case 2;2	n/a	1	1	1	1	1
3	Case 3;2	n/a	1	1	1	1	1
4	Case 4;1 & 4;2	n/a	1	1	1	1	1
5	Freestyle testing	n/a	2	1	2	1	1
	Total	n/a	10				

The log also contains all of my UX testing sessions. An entry into the log for a playtesting session looks like this, referring to my testing protocols and the particular participant:

November 12 '19 - Playtest Session (30 min.)

	Details
	Playtest Protocol 02 - Exploring early levels and character controls
	Session with Sofia Stancheva

Case & Charter Examples

There are templates for both test cases and charters in the Test Plan. I think it worked very well using a standard format for testing and checking.

For the graduation game I had prospected to make at least a dozen cases, but in the end I ended up with just six. The low number of test cases was initially due to the game’s core mechanics still being worked on, and it was therefore difficult to write specific cases; the ever evolving move-set for our character and the levels changing constantly meant that cases had to be updated constantly, which was a bother. As the project progressed the need for more test cases did not arise, and I frankly found it much more intuitive to write a charter and iterate upon it, or to straight up just do freestyle testing, than it was to do test cases.

The rapid iteration on new build also made it very hard to curate test suites. We had planned to do daily builds, i.e. a single, working build each day to test and evaluate, but in reality we had 15+ “daily” builds some

days, which obviously made it difficult to keep up with suites. Putting together a test suite for a build which would only be up to date for 15 minutes just did not make sense. The cases were stored on the same board as the bug tickets for easy access. All six of the final test cases are below as they appear on Trello.com – <https://trello.com/b/LdGEemz1/team-2-buggyboard-ggp>

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

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

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

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

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

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

Charters

I kept my collection of charters in a separate document which made it quick and easy to simply copy and paste existing charters when I need to iterate (This document is also in the Drive folder). I used charters more than cases, but in particular the *freestyle testing* charter, which is ironic, as the idea of charting is to give structure to the exploratory testing process, and freestyle testing is meant as a sort of filler at the end of each testing cycle. I found that the less structured approach often got me to engage more with the game and its systems than specific charters, and especially cases. However I am less confident that I tested all of the game's aspects as extensively as I could have because of it. I have not done quality assurance before, so I have no reference for my coverage, but I can only say I was thorough insofar as I spent a long time with the game.

Charters are meant to be timeboxed, which I also found difficult, as I would consistently just test until I was satisfied with the coverage, or until I was interrupted, rather than stopping after a given amount of time. This was not an issue however, as I was performing the charters alone, and had complete control over my own scheduling. Below are two charters from November 26th (26-11;01) and November 28th (28-11;01), as well as the template I used for new charters:

26-11;01

Mission:	Explore level 0
Scope:	Difficulty, reliability of moving objects, clipping & getting pushed into the map, levers and doors, checkpoints
Allotted time:	20 min.
	2 Music works now, but background audio is broken – the higher the volume setting (in the app) the more the background SFX seem to break. It clips and shit.
	4 Zoe can pull the lever without grabbing it – i.e. touching it without holding onto it – streamable.com/mlgkd
	5 Music works

28-11;01

Mission:	Freestyle testing
Scope:	Whatever
Allotted time:	20 min.
Notes:	1 Collision with lasers in 2.2 again – after dying on the floor-spikes just after the lasers
	2 Zoe can simply run out of lvl 3 – streamable.com/91s74
	3 Accessing the options menu from the main menu can glitch in a variety of ways
	4 Lots of missing textures – on platforms, elevators and "spike-boxes"
	5 Music and sound issues – No music in lvl 2.2 & 3 (except for main menu music of loaded from the level select) – Entering 2.2 from the level selects does not stop the music from the main menu – Returning to the main menu from the pause menu (while in a level) does not stop the level's music or SFX – soundtrack, moving platforms' sounds, footsteps, etc. – If I quit out of lvl 2.1, the level's soundtrack continues to play in the main menu, <i>alongside</i> the main menu music. If I then enter lvl 2.2 (which seemingly has no music) the two music tracks from the main menu will continue to play

DD-MM;II – Template

Mission:	This feature
Scope:	This, this, and this
Allotted time:	X min.
Notes:	1
	2
	3
	4

Bug Tickets & Bugboard

The bugs I discovered were put on a board on Trello.com as separate cards. Below are six examples of bugs as they appear when you inspect them:

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?

Activities

Make Details

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 buggy bois](#)

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
Added 2 Dec at 12:58 - [Comment](#) - [Delete](#) - [Edit](#)

dev_272;02 - we s l i d i n '
in list [New buggy bois](#)

MEMBERS

+

LABELS

Moderate

+

Description

Edit

Build and bug nr.: dev_272;02
Risk (priority): Moderate - 3
Description: Zoe seems to be able to slide abnormally long if sliding under a laser trap in lvl 3, as if the lasers are extending her slide/roll rather than doing damage - see the attachment
Expected result: death i guess
Frequency: 100%

Attachments

LINK

we s l i d i n '
Added 2 Dec at 16:01 - [Comment](#) - [Remove](#) - [Edit](#)

Add an attachment

dev_102;01 - Block missing in lvl 0
in list [Verified](#)

MEMBERS

+

LABELS

Severe

+

Description

Edit

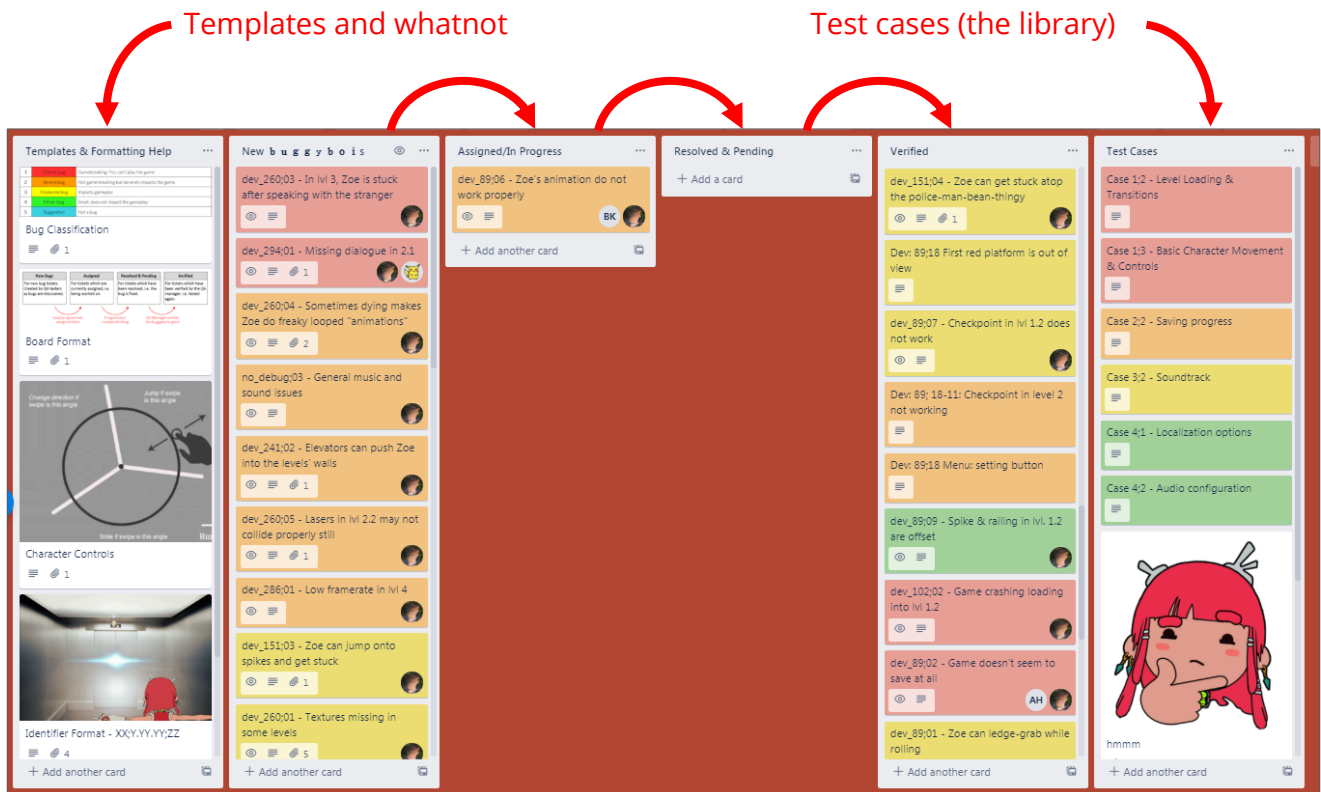
Build and bug nr.: dev_102;01
Risk (priority): Severe - 2
Description: A block is missing i the level, making it potentially impossible to finish. See video.
Expected result: The block should be there i guess
Frequency: 100%
Steps to reproduce:
- Launch the game and open lvl 0
- Progress to the location in the video and observe

Attachments

LINK

https://streamable.com/h18e1
Added 19 Nov at 13:17 - [Comment](#) - [Remove](#) - [Edit](#)

I also include a screenshot of the entire board from December 4th:



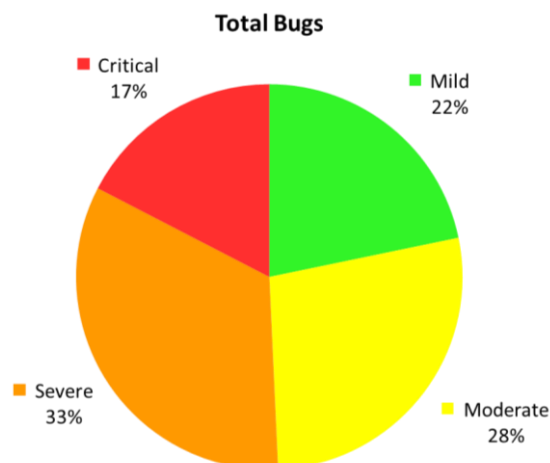
Test Results

I kept my testing log in a spreadsheet, which has made it easy for me to compile and visualize the data into models and graphs. I will evaluate the final game quality based on these models, though I will admit I am not sure exactly what to look for.

Again, there are full-size versions of all of these models in the Google Driver folder:

https://drive.google.com/open?id=1ICZsVqCMYJ_Ev6s3ltvAtM7apwqNUefa

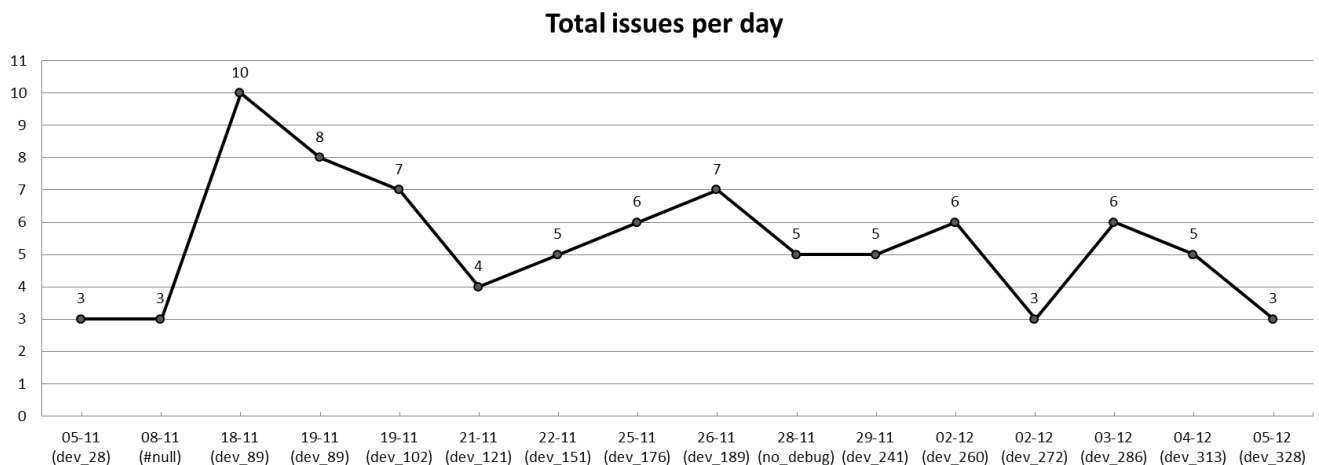
The below chart shows the distribution of total bugs in the four severity levels; I apparently considered a third of all reported issues to be severe, and half to be mild or moderate. I do not consider 17 percent to be a lot for critical bugs, but again I have no frame of reference, so it may very well be. Regardless of what the norm is, 17% of bugs being critical was not a significant hindering, as many of these were level specific, and I could continue testing using debugging tools, or I simply had to wait or a new build, which would be available within a short timeframe.



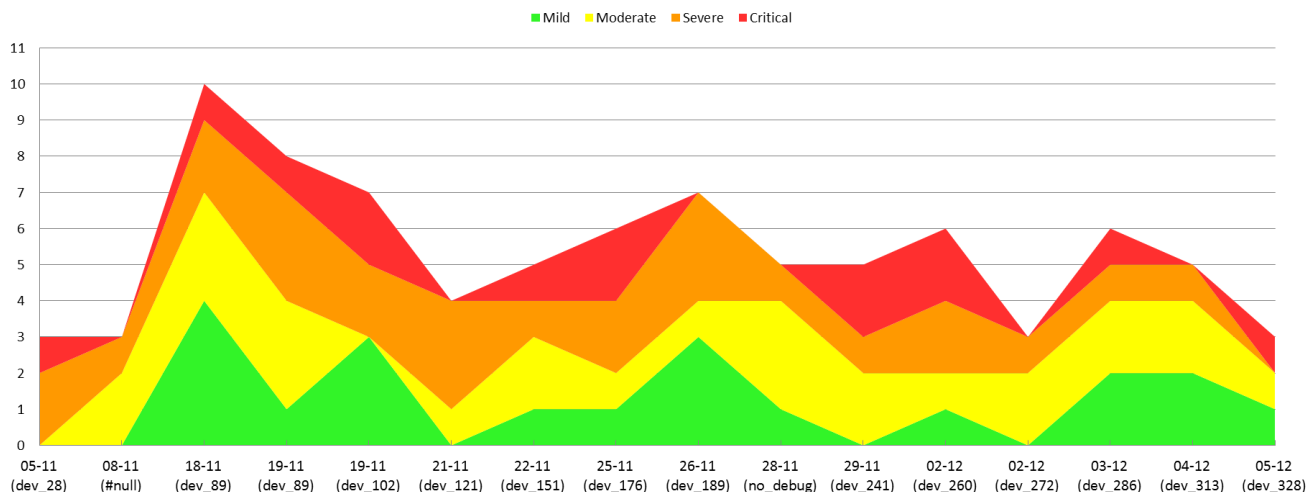
Quality Issues Over Time

I have compiled a graph showing the number of issues found with each session. Because the two first builds to be tested were very early prototypes, it was very limited which aspects I should actually evaluate, which is why there are a few bugs in the first two entries. *dev_89* was the first build to be thoroughly tested, and the graph does indeed show a peak in reported bugs for this build. This is probably also the build which was most thoroughly tested by testers other than myself.

Something else to note is the decrease in bugs over the course of the last three builds. This seems like a good sign, as one might assume that this is because the builds simply have less quality issues, but it is not. As the programmers were preparing for release, there simply was no time to fix all of the minor issues that I might find; the testing I did the day before release had no impact on the game, because the one critical issue I found was already being worked on, and there was no time to look at the rest.



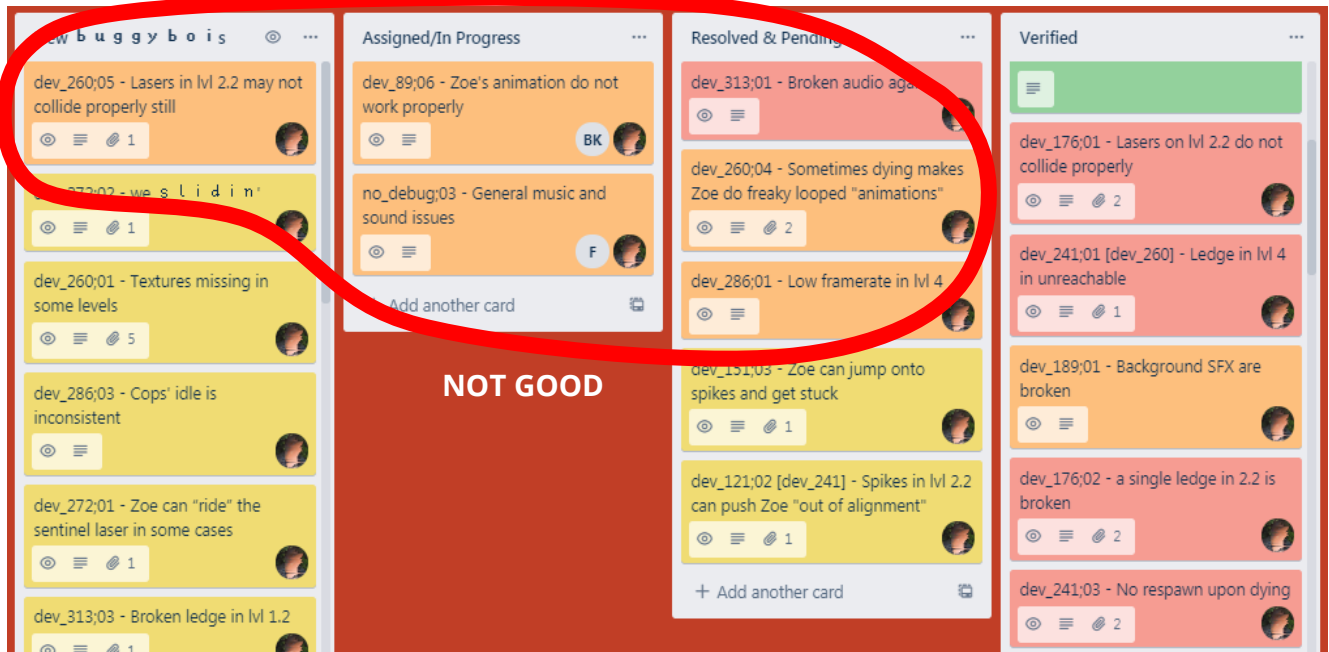
Below here is a more detailed graph, showing the distribution of found issues for each testing session:



Final Testing Result

Seeing as there were still a number of critical bugs late in testing tells me that the game is likely not up to the quality standard that I had hoped for. The test exit criteria (described in the test plan) states that the game should have no critical or severe bugs to be considered *done*. The game *did* have severe bugs upon release, notable collision issues with lasers, animations not playing properly sometimes, and potential audio and performance issues which were never verified.

Below I have highlighted a section of the bug board as it looks on the day of release, December 6th:



The game generally had few critical quality issues, and not definitely no more than I would have expected, but the quality criteria are unfortunately not met.

References

References are here

1. Black, R. (2002). *Managing the testing process: Practical tools and techniques for managing hardware and software testing* (2nd ed). New York: Wiley.
2. Legarth, B. (2019, September 4-6), Three day lecture & workshop on quality assurance at the National Film School of Denmark, Theodor Christensens Pl. 1, 1437 Copenhagen, Denmark
3. Petersen, C. (2019, September 19-20), Two day lecture & workshop on quality assurance at Aalborg University, Rendsburggade 14, 9000 Aalborg, Denmark