Contents

Game Summary 3
Schedule Overview 3
Milestone Overview 4
  Milestone 1 4
  Milestone 2 4
  Alpha 5
  Beta 5
  Final 6
Feature Breakdown 7
  Movement 7
    Left/Right/Jump Movement 7
    Wall Jump 7
  Combat 8
    Melee Attack 8
    Ranged Attack 9
    Thrown Attack 9
  Enemy AI/Art 10
    Simple AI Enemy 11
    Platforming AI Enemy 11
    Flying AI Enemy 11
    Charging AI Enemy 12
    Burrowing AI Enemy 12
    Ranged AI Enemies 12
  Drawing Features 13
    Hat 13
    Melee Weapon 13
    Ranged Weapon 14
    Thrown Weapon 14
    Enemies 14
  Front End/Menus 15
    Main Menu 15
    New Game 15

Load Game 15
Options 15
High Level Schedule 16
Bug Estimates/Projection 18
Bug Tracking ....**Error! Bookmark not defined.**
Staffing and Capacity Plan.. 19
Risk/Issues List 21
Test Script Plan 21
Fresh Eyes/Focus Testing Plan 22
Open for Testing Plan 22
Outsource Testing Plan 22

Game Summary

Page 52 is an action platformer that gives the player the window of creativity to draw and shape their character and their weapons for the story they want to tell. The character has the ability to move left and right, jump, wall jump, sprint, and float in mid-air. All of these abilities are unlocked throughout the course of the game and by progression through the game’s levels. The player is given the ability to draw and create their character’s weapons and accessories to venture through the world with, giving the player a connection to the character that most games lack. Overall, the goal of the game is to jump, draw, and fight your way to Page 52 to rescue the love of your life, Paige.

Schedule Overview

The testing of the game will be heavily done to find bugs in the game’s platforming abilities and in the enemy AI and placement. Much of the testing done is by playthroughs to get the most conclusive feedback on the game and its progress.

During production and early on, the bug testing will all be done internally, while gameplay and design testing will be done externally through classmates and other colleagues. The testing through production (up until alpha) will be done to find major bugs that crash the game and inhibit the production of the game’s core elements. This will also be done for the level editor build by the programmer to make sure that the levels being built will not be made with initial bugs from the engine itself.

Post-alpha, the game will take a stronger focus on the playthroughs as we test for gameplay issues and bugs concerning the AI and the progression of the player through the game.

Much of the time between the start of production and alpha is fairly clear because of the possibility of rush testing needing to be done for the level designs and the mechanics being made at the time. The capacity is done for specific days to show when planned testing will be, although the realistic total capacity is higher. Because of the way the capacity is done, the game will be able to be tested on unscheduled days for design purposes and also some bug testing if necessary.

Milestone Overview

Milestone 1

Deliverables:

- Tutorial Level Designs
- Workable platforming/drawing
- Level Art Style
- Main Character Design
- Prototype Audio/Placeholder
- First Level Art

The bulk of the hours spent testing leading up to M1 will be for the drawing mechanic, and getting it to a functional and testable point for the rest of the development process. The functionality of the drawing mechanic is key for the rest of development so a lot of time will be put into making sure it's not broken and making sure that players who test is will at least be able to enjoy the main feature of the game.

The rest of the time testing will be put towards the platforming mechanics of the game and the level editor to make sure it's at a functional and un-bugged level for the level designer to make (and test) their designs. The platforming mechanics will have already been tested through the prototype in pre-production so the time spent on them will be for measurements and to understand the mechanics. Bugs for the platforming will be documented but not fixed in M1.

The level editor's bugs will be fixed immediately after they're found, as having the level editor is pivotal in the development speed of the game and matching the number of levels the team has set out for them.

Milestone 2

Deliverables:

- First Three Levels Playable
- Platforming Abilities Implemented
- Combat Abilities Implemented

M2 will be used primarily to test levels and the bugs inside of them. Getting our levels to "playthrough" quality is the top priority for this milestone.

Next to the levels, the extra mechanics in the game (for platforming) such as the wall jump, the sprinting, and the floating is also pivotal. The team had issues in pre-production deciding upon which platforming mechanics to input and which to get rid of, so the second milestone will be important in deciding which mechanics and abilities are viable for the direction of the game.

Alpha

Deliverables:

- First Five Levels Finished
- Enemies Polished
- Story and Character Design Finalized
- Final Audio Recorded

The alpha milestone for the team is one of the largest as we plan to release an early-build demo of the game to the public with a small number of levels. The main focus of this is to get the public aware of the game and try out some of the mechanics to see if there's any last-minute changes to the design of the game that need to happen.

To get the project ready for the release, the main portion of the milestone's testing will be done to finalize the mechanics and to test them for any major bugs. Minor level bugs are of a low-level concern at this point and the team will be focusing on getting art into the levels and making sure the character's animations are polished.

The final aspect of the game that needs to be at a viable level for any sort of presentation is the audio, which will get its first big chunk of testing going into alpha. The character's sounds and reactions to the enemies will be the main piece of the testing, while the music will be placed in as a pre-final version to get some feedback. By this point in production, there should be three audio tracks finished for the music as well as all of the character audio finished.

The final audio track will be done in response to the reactions of the public.

Beta

Deliverables:

- Final Audio Tested/Implemented
- Platforming Abilities Implemented
- All Levels Designed

The testing to be done in the Beta phase will be to make sure that the game's core mechanics still function properly with the new levels and the progression of the game.

Also, time will be put towards making sure the menu system is working well and each function in it works.

Final

Deliverables:

- Final Audio In-game
- All Levels Boxed
- All Levels Designed

The testing to be done in the Beta phase will be to make sure that the game’s core mechanics still function properly with the new levels and the progression of the game.

Also, time will be put towards making sure the menu system is working well and each function in it works.

Feature Breakdown

Movement

The team will test these features early on with scripted testing in pre-built levels for the sole purpose of making sure the mechanics work. Later on in the development process they will be tested in playthroughs with many of the other commonly-used mechanics for speed of testing.

The movement features are broken down (and will be tested as) basic movement (walk/jump), and the more complicated wall-jump.

Feature	Sub-Feature	Bug Estimate	Test Time (Hours)	Complexity	Scope
Movement	Left/Right/ Jump	10	4	Low	Low
	Wall-Jump	10	6	Med	Med
Totals		20	10		

Left/Right/Jump Movement

The player has the ability to move left and right using the “A” and “D” Buttons, or the Left and Right arrows. This will move the player off of drops and also into walls, and gives the player air control as well after jumping or while falling. The player will also be able to press “W,” “Space,” or “UP” to jump, and twice to double-jump. This will collide the player with ceilings and walls as well.

Measure of high quality:

- The player will move left and right smoothly with no stopping or degree of acceleration
- The player has full air control as intended
- Control is uninterrupted by other key presses
- The character will jump smoothly as intended with no pausing or glitches in collision

Wall Jump

The player will have the ability to jump off walls in the opposite direction of the direction they jumped at the wall.

Measure of high quality:

- The player will jump off the wall diagonally
- The player won't clip through anything after the wall-jump
- The player will not be able to constantly jump off of the same wall

Combat

The combat features of the game will be tested when they're being built and then finally tested going into the alpha submission. The rest of the testing, and the bugs to be foud afterwards will be found through playthroughs.

The special abilities for each style of combat (not listed) will be tested once they're done and they're in the game.

The combat features are broken down as melee, ranged, and thrown.

Feature	Sub-Feature	Bug Estimate	Test Time (Hours)	Complexity	Scope
Combat	Melee	10	7	Med	Med
	Ranged	10	7	Med	Med
	Thrown	10	7	Med	Med
Totals		30	15		

Melee Attack

The standard Melee Attack works at a set hit-box with a set damage and works with either pressing "Ctrl" or the "Left Mouse Button." The special abilities for it are done through drawing the specified colour in the drawing box for the weapon.

Standard Melee Measure of High Quality:

- The hit-box is consistent
- The damage is dealt to only one opponent
- The animation plays with the attack

Fire Melee Measure of High Quality:

- The attack causes the "fire effect" on the enemy and deals the desired amount of extra damage.
- Units weak to fire will die instantly

- Enemies who are strong against fire take less damage

Stun Melee Measure of High Quality:

- The attack causes the “stun effect” on the enemy and slows their movement by a desired percentage for the desired amount of time
- Units weak to stun will stand still
- Enemies who are strong against stun will not slow down as much

Ranged Attack

The standard Ranged Attack works at a set height with a set damage and works with either pressing “Ctrl” or the “Left Mouse Button.” The special abilities for it are done through drawing the specified colour in the drawing box for the weapon.

Standard Ranged Measure of High Quality:

- The projectile’s hit box is an appropriate size
- The damage is dealt to only one opponent
- The animation plays with the attack
- The projectile launches from the right point

Rocket Ranged Measure of High Quality:

- The attack causes the “rocket effect” on the enemy and deals the desired amount of damage to the enemy hit directly, and the enemies around it
- Units weak to fire will die instantly when hit directly, take extra damage from distance
- Enemies who are strong against fire take less damage, ones who are strong against is and in the explosion radius take no damage

Rapid Fire Melee Measure of High Quality:

- The attack is at an increased speed to the standard ranged attack
- The attack will do less damage than the standard ranged attack

Thrown Attack

The standard Melee Attack works at a set hit-box with a set damage and works with either pressing “Ctrl” or the “Left Mouse Button.” The special abilities for it are done through drawing the specified colour in the drawing box for the weapon.

Standard Thrown Measure of High Quality:

- The hit-box is consistent
- The damage is dealt to only one opponent
- The animation plays with the attack

Grenade Thrown Measure of High Quality:

- The attack causes the “grenade effect” on the enemy and deals the desired amount of extra damage.
- Deals damage to any enemy in range of the hit box

Pull Thrown Measure of High Quality:

- The attack causes the “pull effect” on the enemy and brings them towards the player will dealing damage
- The chain attached to the weapon disappears after coming back to the player

Enemy AI/Art

The Enemy AI in the game will be tested with their art once they’re implemented in a pre-made map for bug testing. Later on they will be tested in playthroughs in packs of levels (along with level art and movement mechanics.)

The enemy features are broken down by movement and then also a section for ranged enemies (breaking down each one) which utilize the movement AI. The reasoning for this is that the abilities the ranged enemies use are all different and breaking down each enemy as their own feature would be too much.

Feature	Sub-Feature	Bug Estimate	Test Time (Hours)	Complexity	Scope
Enemies	Simple AI	5	4	Low	Low
	Platforming AI	5	4	Low	Low
	Flying AI	10	6	Low	Low
	Charging AI	5	4	Med	Med
	Burrowing AI	10	8	High	High
	Ranged AI	5	4	Low	Low
Totals		40	30		

Simple AI Enemy

The simple AI Enemy in our game walks left and right after hitting walls (invisible walls included on edges.) The combat for this enemy is melee, and attacks when the player is within range but will not stop to follow, they only patrol.

- The AI will patrol and not stop when attacking the player
- The AI properly changes direction after hitting walls

Platforming AI Enemy

The platforming AI Enemy in our game walks left and right after hitting walls (invisible walls included on edges) and jumps over gaps. Platforming enemies may also follow the player directly. The combat for these enemies is melee or ranged; melee attacks are done when the player is within range and will continue to attack without moving past the player. Ranged enemy actions are overviewed in the “Ranged AI” section.

- The patrolling versions go left and right and switch their direction after interacting with a wall
- The tracking versions follow the player around the map until hitting walls or being unable to platform upwards
- The enemy will always cross gaps properly and will never die by hitting kill volumes

Flying AI Enemy

The flying AI Enemy in our game moves left and right, or diagonally, towards or above the player. Bees move diagonally towards the player and leave at the bottom corner of the screen. Droppers patrol areas of the maps and drop bombs onto the player. The Diagonal Bombers move diagonally across the screen (bouncing off of walls) and use ranged attacks to hit the player. Ranged enemy actions are overviewed in the “Ranged AI” section.

- The Droppers patrol left and right and switch their direction after interacting with a wall (invisible walls included)
- Bees must hover above the player briefly before going at the player and leaving the screen at the opposite corner
- There should be no collision on the Bees
- The Diagonal Bombers will always collide off of walls and travel on an angle of 45% across the map

Charging AI Enemy

Charging AI Enemy is a specialty melee enemy that attacks the player directly and has increased speed.

- The Enemy will patrol and not stop when attacking the player
- The Enemy properly changes direction after hitting walls
- The Enemy is stunned briefly after hitting a wall

Burrowing AI Enemy

The burrowing AI Enemy in our game travels underground from one side of the screen and follows the player. The combat for this AI is melee, and attacks when the player is within range (above the burrower) instantaneously.

- The Enemy will follow the player underground and attack once the player is above them.
- The Enemy has a ground effect that shows above them
- The enemy will come out of the ground up to player height when the player is directly above them

Ranged AI Enemies

The Ranged AI Enemies in our game fire at the player from a designated distance. The enemies shoot a variety of projectiles.

- The Bow and Arrow enemy shoots in a straight line, with the arrow arching down afterwards. This enemy uses the Platforming Enemy archetype.
- The Man Cannon enemy shoots the player in an arch across the screen towards platforms. The man cannon does not move but can be moved on the ground by the player.
- The tractor beam is a flying enemy using the patrolling AI but uses a ranged mechanic that pulls the player upwards to them.
- Spread shot enemies shoot in a cone pattern in front of them at the player.
- Droppers utilize the patrolling AI for flyers and

Drawing Features

Feature	Sub-Feature	Bug Estimate	Test Time (Hours)	Complexity	Scope
Drawing	Hat	10	5	Med	Med
	Melee	10	5	Med	Med
	Ranged	10	5	Med	Med
	Thrown	10	5	Med	Med
	Enemies	10	5	Med	Med
Totals		50	25		

Hat

The player can draw the hat that their character wears in-game.

- The player should be able to draw what they want on the screen and it will appear on the character in-game
- Colours provided should work properly with the designated density and sizing
- The player can open up the screen and re-do or edit the image they've already drawn

Melee Weapon

The player can draw the melee weapon that their character uses in-game.

- The player should be able to draw what they want on the screen and it will appear on the character in-game
- Colours provided should work properly with the designated density and sizing
- The player can open up the screen and re-do or edit the image they've already drawn
- The player can use special pens to give that weapon and ability

Ranged Weapon

The player can draw the ranged weapon that their character uses in-game.

- The player should be able to draw what they want on the screen and it will appear on the character in-game
- Colours provided should work properly with the designated density and sizing
- The player can open up the screen and re-do or edit the image they've already drawn
- The player can use special pens to give that weapon and ability

Thrown Weapon

The player can draw the throwing weapon that their character uses in-game.

- The player should be able to draw what they want on the screen and it will appear on the character in-game
- Colours provided should work properly with the designated density and sizing
- The player can open up the screen and re-do or edit the image they've already drawn
- The player can use special pens to give that weapon and ability

Enemies

The player can draw the enemy that their character fights in-game.

- The player should be able to draw what they want on the screen and it will appear in the level and track the player
- Colours provided should work properly with the designated density and sizing
- The player can open up the screen and re-do or edit the image they've already drawn

Front End/Menus

Main Menu

The main menu of the game will be tested simply for functionality and to make sure that the animations play properly. The audio for the main menu is tested alongside the functionality to make sure that each action the player can make on the menu has audio feedback.

- The player can click “New Game” and the menu will take them to that screen.
- The player can click “Load Game” and the menu will take them to that screen.
- The player can click “Options” and the menu will take them to that screen.
- The player can click “Facebook” and the menu will take them to that screen.

New Game

The new game menu should be able to put the player in the game from the starting position in the tutorial.

- The player can select “Start New Game” to start up a new playthrough of the game with no crashing or bugs of any sort.
- The “Exit” button should return the player to the main menu.

Load Game

The load game menu should be able to put the player back into their previously played save game.

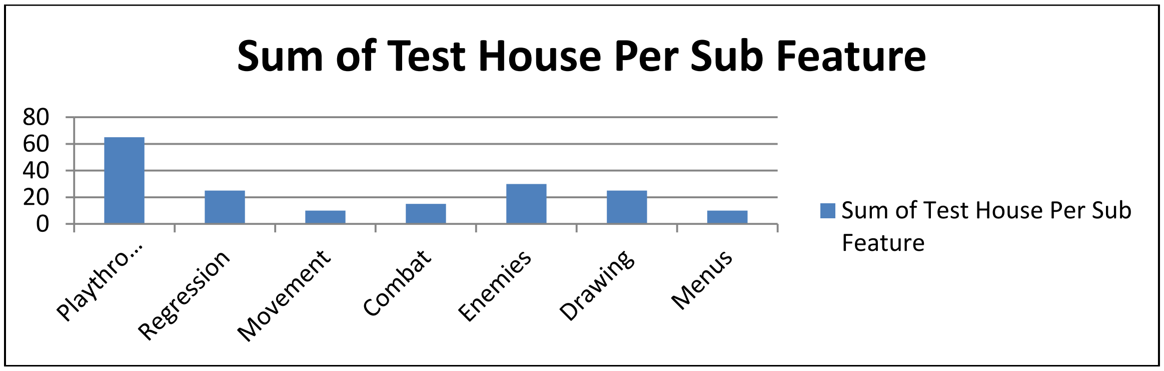
- “Load Save Game” should pull up a list of save game files for the player to select and continue.
- “Exit” should take the player back to the main menu.

Options

The options menu allows the player to change the settings of the game (difficulty, image quality, etc.)

- “Difficulty” should pull up a list of varied difficulties including easy, intermediate, master.
- “Image Quality” allows the player to edit the quality of the game’s visuals if the highest quality should affect the game’s performance.
- “Exit” should take the player back to the main menu.

High Level Schedule

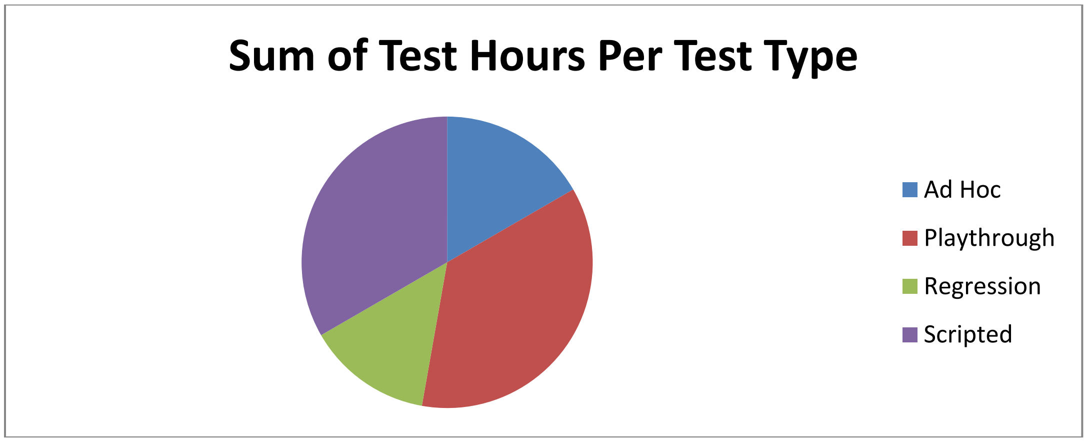


As mentioned before in this plan, the focus of the testing for Page 52 is around playthroughs (post-beta) to get the most consistent feedback on the game as a whole and to find common bugs that the average player will find. Individual testing also takes up a large portion of the testing but is mostly done pre-beta, and is also primarily Scripted to get a strong understanding of how players will use the mechanics of the game in the world we make for them.

Lots of time is put into testing enemy AI as we have many versions of the generalized AI in the game and use them for a variety of enemies. We also focus heavily on the drawing mechanic because it's the most creative and unpredictably used mechanic of the game.

You'll notice there is as much time in Movement as there is in testing the Menus, this is because the movement mechanics we're using are simplistic and individual from the rest of the code, allowing us to finish it early on and polish it before the rest of the mechanics are in the game, so less time is put into testing these mechanics. The combat also may seem surprisingly low, this is because we have only a limited number of weapons, and even though they have special abilities those abilities are mostly cosmetic, as far as the code is concerned. Keeping in mind as well, that many bugs for the combat will be found in just quick testing after tweaking the mechanics, less time will be individually set aside for these mechanics.

Sum of Test Hours Per Sub Feature	
Feature	Total
Playthrough	65
Regression	25
Movement	10
Combat	15
Enemies	30
Drawing	25
Menus	10
Grand Total	180



The test types we're using for our game are Ad Hoc, Playthroughs, Regression, and Scripted. The majority of testing will be done through playthroughs so we can get both balance and bug testing done at the same time in a real environment where people would be playing it. Spending too much time on individual mechanics wouldn't allow for us to test how they work together and what bugs could occur through the progression of our game.

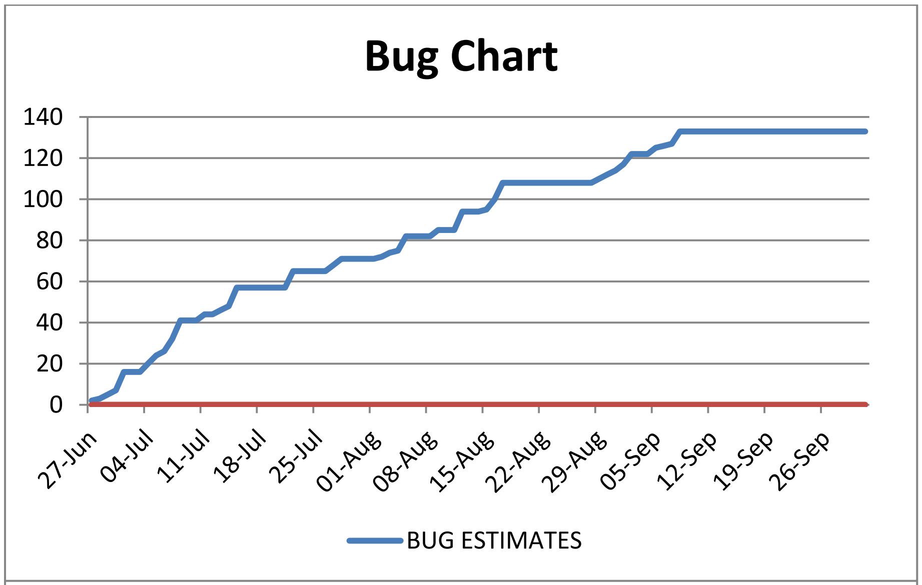
All of the scripted testing we do will be when the mechanics of the game are programmed and it will be done to make sure they're functional, while the ad hoc testing will be done once the mechanics are implemented as they will be part of the playthrough testing.

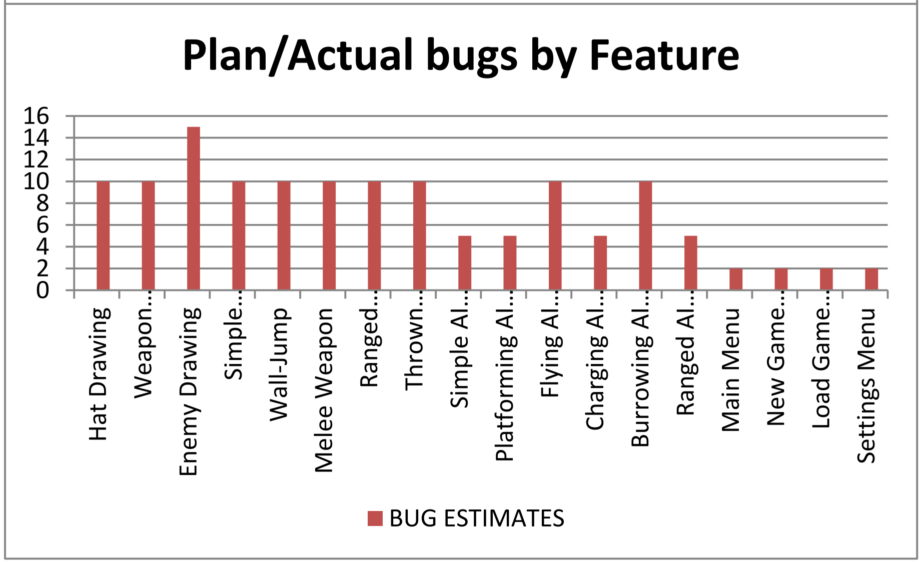
Sum of Test Hours Per Test Type	
Test Type	Total
Ad Hoc	30
Playthrough	65
Regression	25
Scripted	60
Grand Total	180

Bug Estimates/Projection

The bug chart predicts the bug count going up relatively steadily heading right in to break before Beta (17-Aug to 29-Aug) and picks up right afterwards only steadily when final testing begins with playthroughs. Because the playthroughs are also for balancing purposes and not solely for bug testing, we don't predict that we will find too many bugs in that time and that most will be found prior to that. That is also because most of the testing happens while the mechanics are made.

In this bug chart, the "actual bug finds" have not been recorded so there is no second line to show that.





Bug Tracking

The team tracks bugs digitally on excel spreadsheets, placing them in a bug archive folder. The folder is on the school access drive as well so everyone has access to the bug list on campus so anyone can get the list to fix bugs if necessary.

For people testing the game online for us bugs are done through email if player's wish to help with the QA process.

There is no program used besides Excel for the tracking process which makes it accessible for everyone. The bug reports can then also be taken from campus over breaks and weekends if bugs need to be fixed off campus.

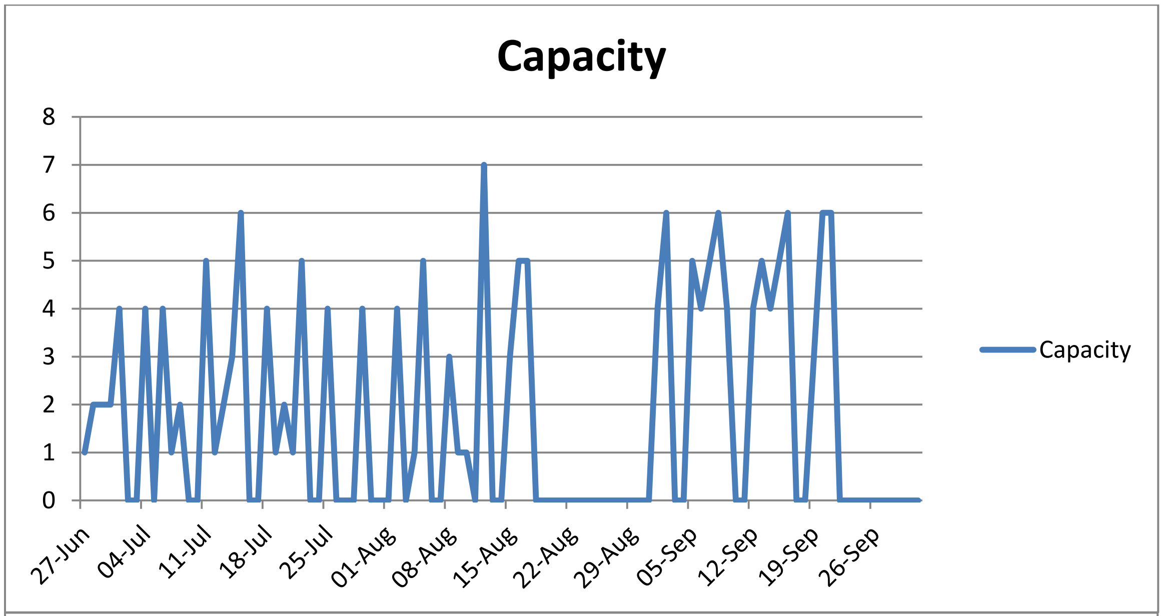
Example of bug write up:

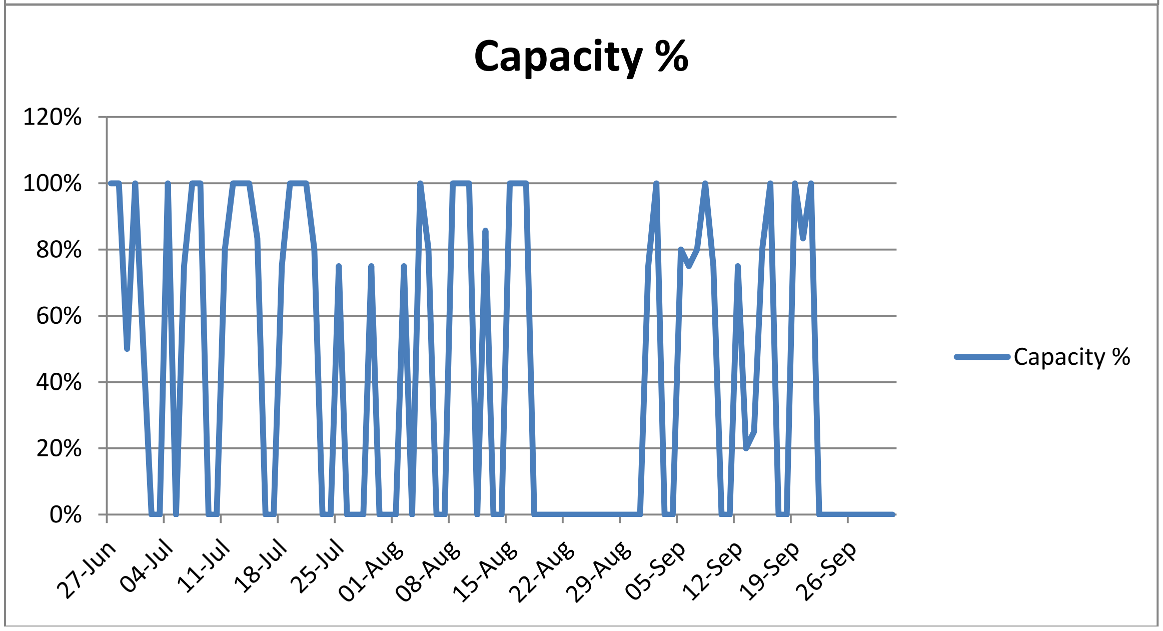
Bug Name	Category	Severity	Rate
Corner Clipping	Movement	High	50%
Comments			
The character clips through the corner of the wall due to the speed of his jump. 1. Stand under corner 2. Jump upwards into the corner 3. Character should clip through the corner of the wall			

We track our bugs through the main features because many of the bugs happen on a wider scope through the code. Many of the sub-features are just small extensions on the main code for a larger mechanic so it makes more sense to track them on only the larger mechanic as they could possibly happen with many of the sub-features.

Staffing and Capacity Plan

The capacity for our team averages out at 85% over the course of the project's development. There are a number of days in the capacity plan where the team is at 100%, this is because with the small number of bugs we predict to find there is a limit on how many hours we'll put into testing for the game. Because there's a smaller number of hours being put into testing, the percentages for capacity are usually at 50%, 66%, 75%, or 100%. Percentages between those numbers are only obtainable in heightened times of testing such as before alpha and in beta. You'll notice the larger spikes in mid-august and in September. Also, in the Percentages Chart there's less flat peaks in the beta phase and at the end of alpha.





Risk/Issues List

There are only a few risks with our project concerning QA:

- The team could fall short on tracking our bugs and resort to simple here-say and close bugs without properly documenting them. This would prove troublesome if we fix bugs multiple times without knowing it and put it too much work without knowing why. Keeping track of the bugs is essential so we can make sure we know where the issues we have come from.
- Because the team is using Flash, many of the bugs the game has must be fixed almost instantaneously because they could stop the game from even starting. This is a problem for us because we could end up only having tracked a small number of bugs over the course of the project because many of them will be fixed before there's even time to document them.

Test Script Plan

The testing we do for our game is mainly to find crash and major bugs to make the game playable to a certain degree. Smaller bugs will be fixed if there's time.

- The tester should only test the feature for the designated amount of time (shown in the test plan schedule)
- Only the features on the list should be focused on for testing. Smaller features should only have bugs found by accident as the limited amount of time we have for testing should be put towards the major features of the game that will always be used.
- Testers shouldn't spend time searching for bugs specifically, they should be found while testing certain portions of the game. If bugs are searched for specifically it should be done outside of the schedule.
- If bugs are found and not documented they should have been fixed immediately. If a bug is found and not fixed it must be documented and placed in the folder on the share drive so it's not overlooked or forgotten about. This will avoid over-documentation.
- Bugs found by testers in the post-alpha release build are tracked via email to a made account for the game.

Fresh Eyes/Focus Testing Plan

The majority of the fresh eyes and focus testing is done by students in the program from other classes. This lets the team get good feedback from students in the Game Design Program and get proper documentation for the bugs found.

As mentioned, the fresh eyes testing is done by the other GD classes. Early on the fresh eyes testing is done by our own class so we don’t have to reveal the game to other classes too early. Once the GD22’s are in the program they will be testing for us, followed by the 21s and then the 20s. We save the 20s for last so we can get the best possible feedback on the game in its pivotal bug testing/fixing period – beta.

Open for Testing Plan

The team will be putting out a post-alpha release build of the game online and asking for feedback on bugs and/or design from the community. This allows us to get unfiltered feedback on the game’s progress and it’s gameplay, as well as it’s bugs.

The game will be released on Newgrounds with instructions on how to submit feedback directly to the team to help us do the best job we can on the final version of the game.

Outsource Testing Plan

The team will be giving the game to testers outside of the school (and not through Newgrounds) to get more solid feedback on the game and get better bug write ups. The people we send it to will be contacts in the industry as well so that we get good feedback on the process of the bug testing.