Review Document

Cat Astro Fee

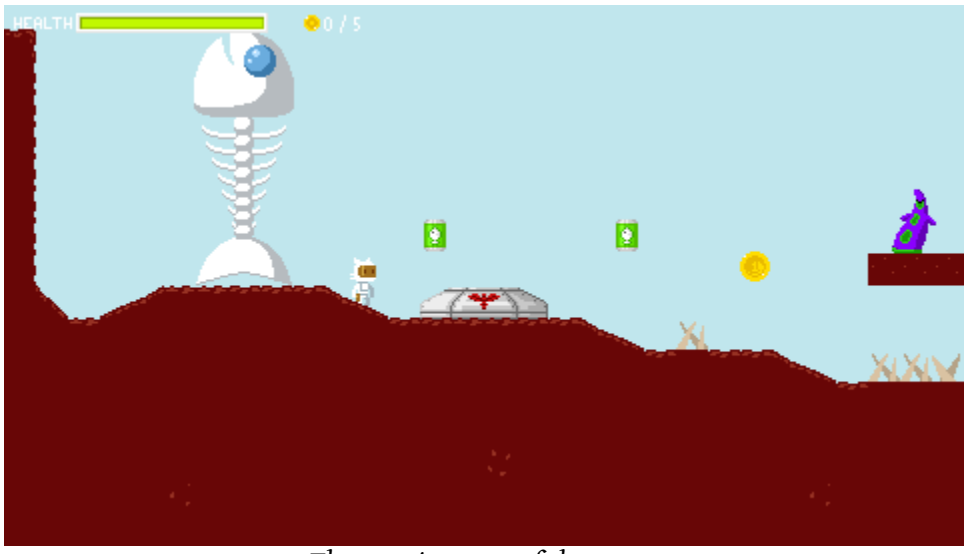
(catastrophe)

Formerly referred to as "Planetary Pitstops"

Nathan Moore
October 23, 2013

Section 1. Introduction

Most of the core game engine features for a side scrolling action platforming game have been successfully implemented within the allotted time frame. This game engine currently is being used for a simple space game starring an astronaut cat who needs to collect enough coins to fly his ship past the space toll booths.



The starting area of the game

Because of the limited content currently in the game, it is much easier than originally intended so it might appeal more to a younger audience than what was in mind when writing the design document. This project has no educational value beyond teaching survival strategies on hostile alien planets.

Section 2. Changes to the Game and Rational

For the most part, the game is largely the same as described in the design document. There are a few differences, though. There hasn't been enough time to implement the jet pack that was going to be used for an additional in air boost for reaching high or distant platforms. There is also only one save file currently available where the design document said there would be three.

Some of the content has been re-themed. The original title, *Planetary Pitstops*, wasn't very good so it was changed to an equally bad new title that has the added benefit of being a terrible pun. Instead of collecting energy crystals for powering the ship the player needs to collect coins for

paying a toll booth fee. The basic game mechanics, however, are the same.

The idea of having the levels broken up into separate areas had been dropped and instead the full planet is available to freely roam. It didn't seem worth it to try to set up regions just for limiting the camera movements and object updating. The original intent of this was to provide a more focused game playing experience in each area of the level, but that didn't seem entirely necessary.

More was planned for the game but time constraints prevented much from being added. For example, a story was going to be placed at the beginning of the game in the form of a slide show. This would have informed the player that the cat astronaut needed to collect coins for the toll booths. It might not be obvious, but to make up for the lack of story some help text can be brought up during play by pressing F1. This tells a little about the story, the controls, and some of the items and objects in the game.

Pressing F1 can bring up some help text

Section 3. Testing Results

Some of the plans described in the Testing Document were carried out, but not all of it. Most of the debugging was done by throwing print statements into the code whenever something was behaving oddly. PyCharm's debugger also came in handy quite a few times when figuring out some of the bugs.

Although I originally intended to use logging and unit testing, those were quickly abandoned to spend more time programming features into the project. In a software project developed over a

longer period of time and with a larger group of people those would seem more necessary, since there is no knowing what one of the other programmers might change and possibly break, but working on my own I can mostly know what to expect.

Debug mode can be toggled with F12

Debugging mode can be toggled with the F12 key, which shows additional information about objects in the scene, like the collision boxes, name, and health levels. This was useful for testing the interactions between objects and their components. It also helps with checking if the components making up the object are properly aligned and correctly sized.

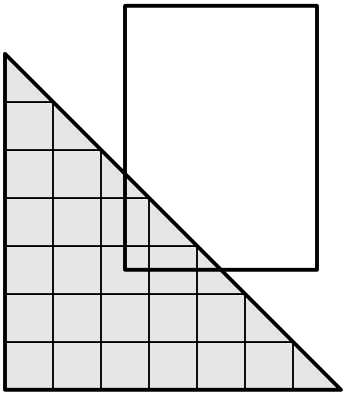
Unfortunately, it was more difficult than anticipated to get people to play test the game. Partly because, for a long time, it was more of a game engine than a game – there wasn't much to play test. There was a lot of testing done, but not by anyone other than myself. There wasn't much yet to get feedback about, since there hasn't been much time for level design or additional content.

Section 4. Technical Issues and Solutions

One major problem encountered during development was with implementing sloped ground. It is surprisingly difficult to find good information about how to successfully achieve slopes considering how many people are interested having this in their games. There are some discussions about some concerns and challenges that need to be overcome, but implementation details are not easy to find.

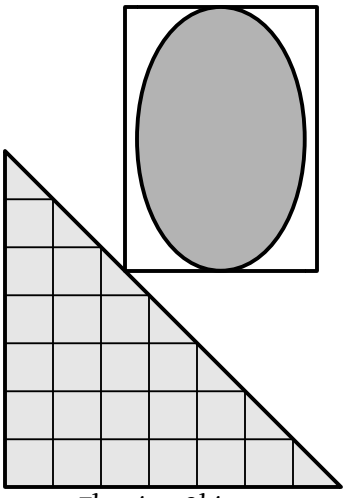
At first slopes were attempted by having the game objects' collision box partially penetrate

the tile map, with the center of the bottom of the box on the ground where the character's feet should be. A problem with this with determining what tile collisions should be ignored, since the box can overlap several tiles. The box will still need to bump up against walls, but not bump against the blocks that are inside the ground.



What collisions should be ignored?

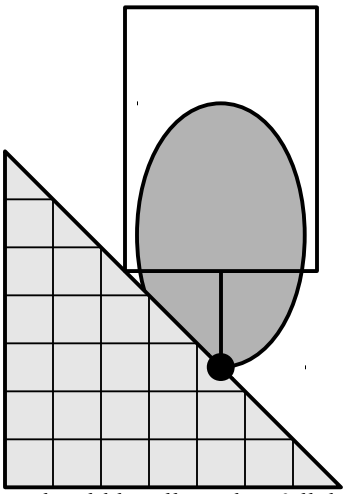
That method was abandoned in favor of not allowing collision boxes to penetrate through the ground. The problem with this is that, when standing on slopes, the character's feet would be floating above the ground (since the box will only go as low as the corner touching the surface.) It works, but it isn't visually correct.



Floating Objects

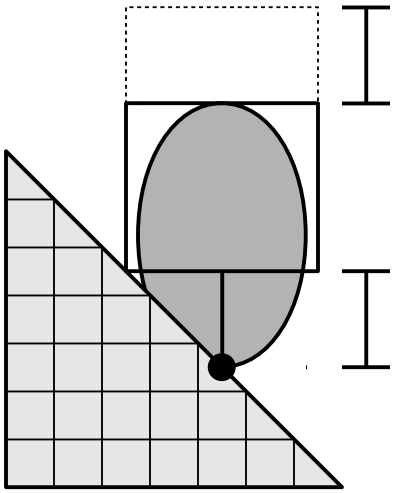
The floating character problem was overcome by testing the intersection between the ground and a vertical line off the bottom center of the box as long as the maximum step down height. If the line intersects the ground, that would be as far as the character will be allowed to fall to. If there is no intersection, then it means the character might be on the edge of a cliff and should only be

allowed to fall as far as the bottom of the box – to walk out off the edge. When the character jumps, it would meet back with the bottom of the box and the collision box and character will move together aligned with eachother.



Line used to test how far objects should be allowed to fall, but collision box is now too tall.

One problem with this is that the collision box will be raised up above the top of the character. This might cause unwanted collisions with tiles above the character that shouldn't be touchable. To compensate, the collision box is shortened by the same amount as the distance between the bottom of the box and the ground point founding using the line test. This could cause a minor issue where, if a character is moving down a slope leading to a block, it could pop up onto the box, since the bottom of the collision box will be above that square tile. This wasn't too much of a concern, so this was simply ignored.



Shorten the collision box by the distance from the bottom to the feet for the correct height.

A potential problem that hasn't been fully tested is if the box is too short or wide, it might try to lower the top of the box below the bottom of it. The collision boxes used in the game are sized in a way where this hasn't been a problem yet, but additional testing should probably be done some time to fix this.

This is the general idea, though the actual code may not clearly reflect that since it was written over the course of changing my mind a half-dozen times about how to implement it. After it started behaving as desired, it was put aside to work on other things. The code for this could use a bit of cleaning up and rewriting to make it clearer how it actually works.

Section 5. Unresolved Technical Issues

There haven't been any major technical issues that have been left unresolved. There are still some features that may have not yet been implemented, but nothing major (a slide show state for displaying the intro story, extra enemies and their behaviors, interactive objects like buttons, switches, and doors, etc.) These were more not done due to time limitations than technical ones.

A minor annoyance that should probably be fixed some time is that, when a level is first loaded, for one frame objects might not be in the desired state during play. This might cause the camera to momentarily be in the wrong place or an object might be showing a frame from the wrong animation. This is more noticeable with lower frame rates since that single frame will be on the screen longer. It may be simple to fix this problem by just calling the object's update function as soon as possible after creation instead of waiting for the next frame. Alternatively, some logic could be put in place to try to have the objects be initialized in a state closer to how they should be during play.

Section 6. Lessons Learned

It probably would have been better to have selected a smaller project – one that could have easily been completed within the time frame. A platforming game engine requires a lot of parts that take time to code. There are also tough content requirements, like a lot of graphics, sounds, and animations. The current state of the game engine is capable of a lot more than what is demonstrated in the game, but the content using it hasn't been made yet – some of the code is not currently used by anything. If there was another month available it could be filled out with more content, bringing

it to a more polished and completed state.

Because of the scope of the project, and other obligations vying for time, there wasn't always a lot of leisure time available. It probably could have been simplified significantly to shorten development time, but that would have limited what could be done with the game engine in the future. For example, mid-level saving could have been dropped in favor of simply saving at the start of the level (and thus not needing to keep track of the player's accomplishments), but this would have prevented larger more challenging levels from being created at a later date.

Also, configurable controls and other optional settings might not have been necessary. There was some feature creep caused by wanting to develop a reasonably flexible system. For a class project, such flexibility might not be necessary.

Section 7. Resources Used

The primary external library used by the project was PyGame. It was used for graphics, sound, inputs, and collision – as one familiar with PyGame might expect. This is a library that has a lot of features that makes game programming easy.

Because the game was designed around the use of Tiled (http://www.mapeditor.org/) for tile map editing, the project includes a copy of tmxlib (https://pytmxlib.readthedocs.org/en/v0.2.0/). Originally, some work was put into parsing the XML documents produced by Tiled, but that was quickly determined to be a tedious waste of time. Tmxlib simplified parsing and using tmx files considerably.

The Gimp (http://www.gimp.org/) was used for drawing the graphics and animations. The sound effects were simply made using Bfxr (http://www.bfxr.net/). Notepad++ (http://notepad-plus-plus.org/) was used for writing the text and JSON files and JSONLint (http://jsonlint.com/) has helped with fixing JSON formatting errors. The IDE primarily used during development was PyCharm 3 Community Edition (http://www.jetbrains.com/pycharm/) and it's debugger was helpful from time to time.

GitHub with GitHub for Windows (https://github.com/) was used for revision control and synchronizing development on both my laptop and desktop. This helped give the confidence needed for checking and rewriting portions of the code. The current state of the project and commit history

can be viewed on GitHub at https://github.com/ArmchairArmada/COS125Project01.

Section 8. Future Plans

The game engine is pretty decent and it would be nice to make use of it in some future project. Likely, if a game is made with it, it will be considerably different from this project with, hopefully, a better theme and less stupid title. Of course, much of the engine code will need to be cleaned up a bit, refactored, and rewritten a little, but nothing too major – just a little house keeping.

For a while after the deadline, though, nothing will be done with the project. A break from it is needed and some time should be spent getting caught up on things that might have been a little bit neglected. It would be fun to return to it some time and maybe get some help and feedback from a few other people with making content.

Section 9. Bibliography

[1] Rodrigo monteiro (May 20, 2012) "The guide to implementing 2D platformers" [blog post] Available: http://higherorderfun.com/blog/2012/05/20/the-guide-to-implementing-2d-platformers/

[2] "TMX Map Format" [wiki page] Available: https://github.com/bjorn/tiled/wiki/TMX-Map-Format

[3] "tmxlib: the Python tilemap toolkit" [documentation] Available: https://pytmxlib.readthedocs.org/en/v0.2.0/

[4] PyGame Documentation [documentation] Available: http://www.pygame.org/docs/

Section 10. Work Assistance Statement

The work done on this project and review document was all done by me. Although more feedback and ideas would have been appreciated from more people, it seemed more difficult than anticipated to find the time to demonstrate what I had made to others.