Comp Sci NEA Analysis

Introduction:

My project is a “speed-runny” based platformer in which the player – the “Star” – will initially race through a level of platforms and other obstacles, with a single goal of reaching the end as fast as possible. After the first run through, the player’s “Shadow” will follow them, but unlike certain games like Mario Galaxy, where the player’s shadow follow their movements with a 3s delay, this player’s shadow will recreate the exact same path as the prior run through, either will recreate the player’s inputs or will use the player’s position data from the prior race to recreate the movements in a way. The overall aim is to beat your previous time like most speedrunners do in their respective games, looking for millisecond improvements in places others wouldn’t think to try.

The basis of the movement is going to be similar to that of “Dead Cells” by Motion Twin, with its very fast horizontal movement and some movement aspects from that of “Haste” by Landfall. Although haste is a 3d game, its slope acceleration mechanics would be a good feature to try and implement. Dead Cells is a good example of movement mechanics as along with its movement being nice for general use, the devs used it for things like hidden parkour challenges that test the player solely on their movement capabilities.

Computational Methods Used:

The main reason that I am using a computer to solve this problem is that a game of this design isn’t physically possible to create, either as a tabletop game or one played in real life like tag, this is because of quite a few reasons, one example being that the “Shadow” will be a perfect recreation of your past movement, and though we have the methods to get the exact same time scale, the general chaos that happens throughout the real world makes it physically impossible for a prefect recreation of the exact events, thankfully, a computer allows for us to mitigate the chaos of the world around us. On top of the innate chaos of the world around us, our world is 3 dimensional and because of this, my 2-dimensional game would be borderline impossible to create physically as keeping something to just 2 dimensions is hard. The other thing is that my movement is going to be ambitious, with no speed limit for the character, this is something not possible in real life due to terminal velocity existing.



Audience and Stakeholders:

My target audience for this is people who enjoy platformer games like “Hollow Knight” and “Dead Cells” and also enjoy the repetitiveness that these games can bring, with combat devoid gameplay and much more movement based than some of the other types of platformers out there.

My Main stakeholder is my friend Joe Reynolds; this is because he has a similar taste in games to me and has agreed to help with my project by giving advice on what to change and/or what features to add. He will benefit from being a stakeholder as he has a knack for video games and with the school’s limitations on playing games within school, having something he can play without it being blocked would be useful to him. As my main stakeholder doesn’t have a computer at home that I am aware of and that is capable of running my game, He will most likely be playing it on a school computer. The Specs of a school computer are as follows: 1920x1080 monitor, intel core i5-9500, 16 GBs of ram, integrated graphics and 256 GBs of internal storage.

The main inspirations for this project are as follows:

Super Meat Boy:

a similar platformer game by Edmund McMillen in which the player plays as meat boy and intends to save bandage girl from the many traps and other obstacles set around, includes a list of selectable characters with different traits and gameplay styles. The current \*any% world record is **17m 27s 267ms** by Matte, for reference, the second closest is 283ms slower, just over a quarter of a second.

Trackmania:

a racing game in which avid players strive to perfect the maps that they race on, aiming for a perfect time with the aforementioned millisecond improvements in the tiniest input differences, sometimes borderline impossible for humans to achieve as they may be only 1 or 2 frame windows (1/60or 2/60ths of a second for most players) Trackmania doesn’t necessarily have an \*any% record as there are maps that the players can choose to play on.

On the right is an example of what is called a “ghost” in Trackmania, this is a past race someone has ran on the same track that the player is currently racing on, this person may be the player themselves past run, or a world record for a track.

speedrun.com:

the home of speedrunning, where all the speedrunners will upload and view everyone else’s times, all games with runs uploaded for them will be listed here along with their categories and sub-categories

\* any% is the typical speedrunning category for borderline all games and constitutes simply beating the game as fast as possible, by almost any means possible, the “almost” is because most games are ran with the “NMG” or “No Major Glitches” ruleset, meaning game breaking bugs that can get you to the end of the game within minutes. Some games do allow this like The Legend of Zelda: Ocarina of Time, where in the world record, they use a glitch to warp to the end of the game, beating it in around 3-4 minutes.

Dead Cells:

Basic Movement Mechanics:

Dead Cells has very snappy movement that is responsive and fast, the player has quite quick acceleration, but you can avoid that with its rolling mechanic, I aim to recreate the movement in some effect, with at least the acceleration and the relatively quick speed. I do plan to have a “soft velocity cap” in which if you are going too slow, you will accelerate to reach it, but if you are over the limit, you will have a slowly increasing deceleration that will return you to speed. The progress of the deceleration’s increase will be reset in a manner of ways, like if you are maintaining a higher than cap velocity by constantly dashing, it won’t be able to begin properly decreasing the velocity. This is similar to Haste in a way as there is a lot of accelerating down slopes and you are given a velocity meter, that shows your current speed.

Haste:

Acceleration/ Deceleration:

As previously mentioned, I want to implement some form(s) of acceleration similar to what Haste has, this includes, basic acceleration up to an exceedable limit, this will require, the player to have a variable velocity, which at the current time is implemented, a soft velocity cap, which I currently haven’t implemented, and a few different forms of acceleration, these will be, on ground acceleration, which will progressively add to the players velocity up to the soft cap assuming no other acceleration is used, item acceleration, which will either: temporarily increase the soft cap so the player can go faster on average, or, add a large number to the players velocity but will also immediately begin max soft cap deceleration\*

Another Feature I think would be a good addition is the item system, you have three types of items, repeating, triggered and active. These will, activate on a repeating timer, say, once a second, activate when meeting a requirement, such as making a perfect landing, and activating when the player presses the button to activate it. I find that this would be a good system to add, with having special stages specifically for items between the levels, but unlike haste, in which you buy them with a currency, you would have a choice of one or two from between 2-5 items, either based off of performance or chance

Ultrakill:

Replayability:

In Ultrakill, when you beat a level, you get given a ranking that is based off of 3 categories: Time Taken, Kill Count and Score. Each category ranges from D-S, and they will combine to give the player their final rank. The reason this is a feature worth mentioning is because there is extra content locked behind the player beating every level Perfectly, and in doing so, obtaining a P rank on that level. This increases the replayability of the game massively and therefore is a good example of why I want the Shadow to be a feature, as it will encourage the player to replay older levels to try and get a good score.

\*Soft Cap Deceleration is a feature I plan to implement to counteract the players speed going too high, how it will work is, when the player’s velocity goes above the soft cap, a separate value will begin to increase over time as long as they are above the cap, this value is affectively an accelerating deceleration. Assuming the player is above the soft cap, the SCD value will begin increasing, progressively slowing them down more and more, until they are pushed below the soft cap, where SCD will begin decreasing rapidly, not instantly to avoid some buggy abuse of the feature.

Alto’s Adventure:

Alto’s Adventure is a 2d snowboarding game with very smooth feeling movement and cozy gameplay. Its main standpoint for me is the acceleration gain the player gets when doing “tricks” among other things. This would be a good feature to add but also this would need tricks which would be a feature I’m unsure about adding. This is just a point that came up because I felt the need to point it out.

Hardware and Software Requirements:

Device: I plan to have the game played on a computer and not a console, so any computer capable of running a game through pygame will be enough. The reason I’m not porting my game to console is because it is beyond the limitations of my time and capabilities for this game. This is because I am unaware on how to port it to console in the first place, getting the game onto the store costs money and I would have to spend extra time programming more stuff like controls and the menu functionality. The device running my game will need a varying amount of memory, for the windows computer I am typically running it on, they both have 16 GB of ram and use about 8 actively, but if my game was ran on a less intensive OS like Linux, it may need less ram.

Controls: I am going to code the game such that the main way of playing it is on a qwerty keyboard. This is because it is the standard in my country, and I don’t plan for people outside the country to play my game in the first place. The reason the controls will be specific to a qwerty keyboard is because some keyboards have the keys that I am using for movement in different places. Among the other binds. A mouse isn’t going to be necessary as

Display: My game has code that will auto fit the game to the resolution of the monitor, this means that theoretically it could be played on a monitor any size but for playabilities sake, I intent to have my game played on a monitor that is equal to or above 720p. This is so my game is at high enough of a resolution to be visibly good looking.

OS: As my game will be ran through a code compiler or an ide like visual studio, the computer will need an OS capable of running the necessary software. This will work on Windows as it is the OS the game is built on, but others like Linux and MacOS should be able to run it as well, assuming they can get the correct software. I am slightly unsure on what will be able to run my game other than windows as I am unexperienced with them and I am unaware of their capabilities beyond some basic knowledge from friends who use them.

|  |  |  |
| --- | --- | --- |
| Features | Description | Why it Is / Isn’t useful to the game |
| Basic Movement Controls | The basis of the movement in my game, holding a/d will cause the character to move left/right respectively, pressing space will cause the character to jump, holding w near a ladder/rope equivalent will cause the character to climb it. | Essential – The movement will be the basis of the game and without it, the game would be lacking its core function. |
| Methods of acceleration/  deceleration | When holding a movement key, the player will begin to move in the corresponding direction, i.e. left or right. Instead of having the player reach speed instantly, I will add acceleration to the player’s movement. This will tie in with the other methods of acceleration in the game, like how falling will cause the player to accelerate, as will going down slopes. For the sake of not having uncapped speed, I will implement a speed “soft cap”, this means that is the player is above the soft cap, they will slowly begin to decelerate back to the soft cap. This deceleration can be counteracted by gaining some form of acceleration, say, using an item or going down a slope. Item acceleration will be another way of gaining speed, it will work something like this, either, the player gets an item that, upon use, causes the player to gain a large volume of velocity but also immediately max the “Soft Cap Deceleration”, or the player gets a passive item that incrementally increases the “Soft Cap”, say going from 80 to 85. | Essential – this is to make the game feel smoother and less like the player is going to be stuck at a constant speed the whole game. Imagine a race but all the cars can only move at one speed the entire race, it just wouldn’t work very well. This goes along with the fact that one of the main games I’m basing my work off of is almost entirely based around acceleration. |
| The “Shadow” - The Player’s Past Run through. | After completing a run of a map, and finishing the final stage, the player will be made to play through it again at least once before moving on, this is so the player has a chance to play with the “Shadow”. The shadow is a recreation of the player’s position data from their first run of the map. The point of running it back is so the player can play against the shadow and try to do better than their past. | Essential – this is another one of the game’s main mechanics, as it is what encourages the player to try and beat their previous times instead of just playing a level once and moving on.  This ties in with the fact Ultrakill is a game I take inspiration from surrounding replayability. |
| Cont. - Basic Acceleration | The Basic Acceleration the player will experience will be based around the soft cap for their velocity. If the Player is below the Soft Cap, they will accelerate up to it. When going down a slope, the player will accelerate beyond the soft cap. | As this feature is part of the core movement system, it is essential, and I am going to add it to the game. |
| Cont. - Soft Cap Deceleration | The Soft Cap for player speed is going to determine how fast the player moves, typically, this soft cap is automatically implemented through friction, which is one way I could do add this function into the game, but I’d prefer for it to act as some sort of curve, where the amount it decelerates the player gets more over time, or up to a max instantaneously through use of an item. If I were to implement it without using friction as its basis, I would have no friction during movement, which would typically be a problem, but I have thought of ways around it, say, having friction as a feature, but only having it active when the player isn’t holding down a movement key. This would lead to a problem however, as if the player got up to speed going one direction, and in the next frame, swapped from holding left to holding right, the player would still have a good amount of momentum towards the right, and this would lead to some form of slipping. Another way would be having instant stopping but that makes the game feel clunky. | With the amount I have gone into this feature, I definitely want to add this to the game, therefore making it essential to the game’s function. |
| Items and Item Based Acceleration | Though I don’t have a definite list of items at the current moment, I have some general ideas. I plan to add 4 types of items, similar to that of Haste, there will be passive items, that act without a condition and apply and effect when the player has it in their inventory, active items, that will cause an effect on the player when a specific button is pressed, and triggered items, which go off when a certain condition is met. This condition can be anything from saving you when you fall or when finishing a level. The final item type is repeating, this will trigger on a timer, such as giving the player a tiny boost once every second or something of the like. | This is a feature I’d like to add so for now I’m going to set it as a primary task that will be added to the game but if the time comes and it’s too hard to make or will take too much time then my plans may change. |
| Movement Methods cont. | There are a few more features surrounding movement that I think would be good to add but I am unsure on, first Is wall-jumping, in which the player would be able to optionally jump off of a wall, turning all their movement speed in one direction to the other direction. I think I could facilitate this by allowing for a wall grab feature, where the player will have a designated period of time to hold on to a wall and decide whether to jump or not, this feature will be broken up into 2 parts, the initial grab, lasting a small amount of time, in this period, if the player does a wall-jump, they will leave the wall with the same speed that they hit the wall with, or if that speed is lower than a certain value, they will jump with that said value of speed, the other part is the “slide” part, in which after holding onto the wall but not jumping, the player will begin to slide down the wall. Any wall-jump initiated during this state will leave the wall with the previously mentioned set speed. This allows the player for some more control over their speed and how they move. Another movement feature id like to add would be a ledge grab. This would allow for the player to be able to make some jumps that wouldn’t be possible without it. Though I think it would be a good feature to add, I do think it would be a challenge and on top of that its not completely necessary. |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

