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.



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.

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.

|  |  |  |
| --- | --- | --- |
| 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. |
| 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. |  |
| 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. |  |
| Items and Item Based Acceleration | Though I don’t have a definite list of item 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. |  |