BeatNinja 3231 GDD

Evan Scott-D'Souza, c2030044

Project Repository: https://github.com/EvanWSD/BeatNinja

Prototype Demo: https://youtu.be/mfi79AmjV2Q

Concept

A skill-based movement-fps in the wake of "Ultrakill" that weaves exploration and challenge into a web of interconnected levels with a focus on replayability – all to a beat.

Player Experience

Play Style and Subjective Experiences

Levels are built to be learned and mastered – unlocking opportunities for additional explorations and challenge. Playing a level for the first time will likely be a player's worst attempt in a similar manner to how one learns to walk – by first learning how to crawl, then walk, then run.

These different experiences therefore emerge in different playstyles – one for learning levels and becoming comfortable with running it, and further mastering it - being able to complete it through muscle memory. A great metaphor is that of learning to play a song using an instrument, and then years later being able to play that same song without any thought put into it. This is also why there are two playthroughs of the level in the provided demonstration video – once normally and once with the expected skill and exploitation of the core mechanics that allow the player to reach the challenge door in time.

The player should always feel like losing is their own fault and never the game's – a distinct point from where the fault is in reality, which is no more than a guideline.

The game should appeal to the speedrunning community (or those of difficult games such as Soulslikes) through generating a similar atmosphere to the feeling of being on a run that is on pace with a new personal best. Upon failure, the player can retry almost immediately to encourage them to rack up a number of attempts before success without dwelling on their failures too much.

Player Taxonomies

A player-centric design approach was taken and therefore intends to adhere to the following 'classes' of player:

- Daredevil – the game is fast paced and challenging. Good runs have value due to difficulty and therefore pressure is easily built up, prompting players to fight off any added stress.
- Conqueror – winning is hard and should take several attempts thereby increasing the psychological reward, especially as part of the positive feedback loop created by one hit failure and additional challenges to levels on repeat playthroughs.
- Mastermind – finding alternative exits and figuring out how to get to them with enough skill and precision is required in order to experience most of the game.

MDA Framework

Mechanics (more detail below)

Most abilities require input to be timed with the same beat that the level moves to. The player is highly mobile with a double jump and ability to dash into enemies or their bullets which gives them an upwards impulse. They also have a bow that can be used to hit buttons in the level or to ignite midair grenades – the latter of which otherwise explode after 4 music beats, either way defeating nearby enemies. Levels are split into checkpointed sections with goals that deny progression until met such as defeating enemies or

hitting buttons. Furthermore, levels contain challenge doors that serve as alternative routes to secret exits that can branch to different levels like speedrun challenges or no-death runs. Bullet hits or falling off the map are instant death for the player.

Dynamics

Thus, the player must both plan, improve and execute their strategies in a rhythm such as "pull bow, dash bullet, shoot button, dash bullet, pull bow, throw grenade, dash grenade, shoot grenade" – with each action completed on a different beat. The prototype level exemplifies this in figure 4.

Furthermore, this rhythm almost becomes its own song that is learnt through repeated practice of the level, eventually becoming muscle memory and encouraging the player to enter a flow state, and eventually they are rewarded for perfecting these mental scripts.

Aesthetics

The player's emotional result should be reflective of that of a speedrunner's, despite not always having the same goal. The player should feel like they have a choice between running headfirst into challenges and beating them through repetition and getting a good run, or instead to explore enough to find a different challenge somewhere else in the web of levels that might help them instead.

Interestingly, this further creates two distinct positive and negative feedback loops. The positive loop comes from learning a level, where failure is common and punished by the player restarting from a checkpoint and rewarding their level completions with progression to later levels and new abilities. However, the negative loop then appears due to how a skilled player can either take on a harder challenge with less abilities, or they can concede to a different challenge where they can get an ability that would make the first challenge easier, which in turn helps lower-skill players. This lowered skill floor and high ceiling culminate in giving the player the ability to guide themselves into their own cathartically enjoyable difficulty (where they are most likely to enter a flow state) without any discrete setting or dynamic enemy behaviour.

On a shorter-term basis, given that almost all elements of the game will be responding the music in some way, any negative response to failure should be consciously and subconsciously dampened by its rhythm for the same reasons people listen to upbeat music in the first place – it hypes people up.

Game Mechanics

Rhythm

Each level has music that plays as soon as the level starts and loops afterward. Certain mechanics (described in more detail in their own section) are timed to the music's beat using a known bpm.

Some player abilities will not properly activate if improperly timed with the beat. In the prototype, the ability will not activate at all, though the following ability descriptions include "half-activations" to communicate response to the player without properly applying the ability thereby punishing the player for not playing to the music.

Dash

Upon player input [LSHIFT], the player quickly and momentarily dashes forward in their facing direction, ignoring gravity and previous velocity. At the end of the dash, the player's velocity is zeroed out. Enemies, the bullets they fire and player grenades are labelled as valid dash targets. If the player hits a dash target, a dash is known as "successful" - the dash will end early and the player will be impulsed upwards. The impulse strength should be greater than the jump and double jump's impulse. The ability to dash refreshes at the same time as double jumps – on successful dashes and on being grounded. If the input is mistimed, the player dashes for a small fraction of the usual distance and loses their charge. Figure 3 exemplifies an intended use case that is expanded on in the prototype level design section.

Double Jump

A simple double jump that refreshes a charge upon being grounded or after hitting a successful dash target, using Rigidbody impulse.

Bow

Hitscan bow in the hud that can be used by the player to hit certain objects for different effects: buttons will activate and increment button section progression and/or open a door if attached to one. Furthermore, grenades will explode early upon being shot.

To fire the bow, the keybind [LMB] must be held down for at least half a beat, and then released on a beat. If either of these requirements are not fulfilled, the bow will reset without firing. The player can keep the bow pulled indefinitely even while dashing and throwing grenades. Enemies are unaffected by the bow to encourage players to risk either moving close to adversaries or the grenade skillshot.

Grenades

Grenades can be thrown by the player if they input the keybind [F] in time with the music, or else the grenade will not be thrown. Once thrown the grenade will detonate after 4 beats of the music or after being shot by the player's bow. On detonation, the enemies within a small radius will be eliminated, though the player will be unaffected – see figure 2. Since grenades can defeat multiple enemies at once, they have an 8 beat cooldown since they would be too powerful if usable on every beat.

Grenades have 3 modes:

- Icy – will slide along the ground
- Bouncy – will slide along the ground
- Gravity – does not abide by gravity until the grenade keybind is pressed again, upon which the grenade is strongly affected by gravity as if it became extremely heavy, and detonates upon hitting the ground.

The grenade is also a valid dash target – it will not explode but will apply the standard upward impulse to the player upon a successful dash.

Sliding

When dashing into the floor, the player can hold [C] during the end of the dash to maintain some of their momentum until they let go of the button and return to normal. The player's speed will naturally decrease over time due to friction on the ground. The player can maintain pulling their bow during slides. The immediate speed upon beginning a slide should outspeed normal walking.

Level Sections and Checkpoints

As mentioned in the level design section, levels are split into sections that deny progression until that sections's prerequisite is met, which is defined on checkpoints that start a new section (some checkpoints are placed mid-section and do not begin a new section, though they will update the player's respawn position. Checkpoints will only update the player's spawn position if the player's current checkpoint index is less than the checkpoint's.

If the player falls of off the map into death plane or is hit by an enemy's bullet (when the prototype's godmode is not enabled in the main menu for the prototype), they will restart at their last reached checkpoint.

Low Gravity Powerup

Small, purple cubes denote powerups that will change the global gravity of the level for a short time (around 10 seconds) – with some additional randomness such that the time slightly varies thus prompting the player to improvise on when it ends and further adding variety to repeat playthroughs.

Predicate Doors

Another feature greatly expanded on in the subsequent level design section, levels have multiple routes to take and some routes are blocked behind doors that open or close depending on a challenge like reaching it without being hit or within a certain time thereby prompting the player to master a level to be able to access all of its routes and therefore reach all of its potential exits on different runs. The time limit for the speedrun-related doors will be found through timing how long it takes a skilled player to reach the door with some added leniency respective to the intended weight of each independent challenge. Furthermore, after the game's release, specific challenges could be buffed or nerfed in updates depending on the

emergent average skill of the consumer playerbase and how this differed from the internal playtests described in this document.

Musical Block Systems

Colored level geometry that alternates being active by color so that one of n colors is active at a time. For example, if a system has 4 colors (e.g. blue, pink, green and yellow), only the blocks of a one color will be active between two beats, and then the next color in the sequence will be active on the next beat and so on (the colors will also wrap from the last to the first).

Enemies

Enemies are able to fire slow-moving projectile bullets (that are valid dash targets, and move faster than the player's default walking speed) in time with the music. Yellow shooter enemies will not move and will shoot on every fourth beat. Pink mover enemies will shoot on every beat and will move with more complex AI that alternates between moving offensively and defensively (towards and away from the player). If the player moves too close to the pink moving enemies, they will back away while continuing to shoot at the player.

Switcher enemies work with the musical block system, to only be able to be attacked on every n beats and further would be colored to signify to the player at which beat they are vulnerable.

Structure / Level Design

Overview

Game split into three areas split into a branching web of levels comprised of multiple sections.

General Level Structure

The game features a "mesh" of levels, akin to a directed computational graph, traversed by completing levels with more than one possible exit. Alternative routes can be found through exploration and as rewards for high-skill or highly-progressed players (such as a door to a secret exit that only remains open if the player gets to the door without dying or before a certain time limit). Figure 1 demonstrates a design for the first area of the game. Nodes are levels and edges are transitions between levels. Labels within nodes are either abilities gotten within the level or a description of the level itself whereas labels on edges are the intended requirements for the player to reach such a level transition.

On level select, players will be able to select any level/node that is adjacent to a level they have previously reached an exit on. This means that players will need to replay levels to see all of the game and therefore reinforces the intention that they in turn practice these levels and eventually enter a flow state, potentially building some muscle memory for the different sections.

The location the player begins the level in will depend on how they got there – which means that replaying a level after getting there from a newly discovered route may start the player in an entirely new section of the level. The resulting path of the player may cross or even overlap with the path they took the last time they played, depending on their skill and unlocked abilities.

Some of the 'timed' secret exit doors will require an alternative route that would not be accessible on the player's first journey there – intended to invoke creativity and imagination in how the player approaches reaching these alternative doors, since they are likely to happen upon closed doors that clearly would have led them to a more 'prestigious' route, leaving them wondering how to get through on a subsequent attempt at the level. It should be noted that the precondition for these alternative section doors will be communicated by the doors design – timed doors, no-death doors etc will be textured differently. Some alternative exits will require abilities that won't be obtained until later levels thus prompting players to return with more context.

Some levels will have different music with different rhythms depending on the entrance the player used to reach it. For example, if the player finds a novel way to return to an early level (such as entering level 4 through level 6), the song would be different to the usual, with a higher bpm resulting in increased difficulty, but allowing for the player to reach a timed door sooner.

A subset of these “predicated” exits will be technically possible through mastery of the limited mechanics at their disposal, although this won’t be made obvious by the level design and instead will be designed in such a way to seem unintentional. This will act as an extra challenge for higher-skilled players and will reward them with a shortcut through some of the easier earlier levels. The subsequent alternative level would therefore not strictly require the use of the ability that most players would have used to get them there, and its completion without it would serve as additional challenge.

Levels are each played in subsections (divided by large doors) that each give the player a goal to complete such as eliminating a certain number of enemies or hitting enough buttons. Subsections are typically encountered in order and the level design should allow for each subsection to lead naturally onto the next upon completion. Some levels will allow leniency in the order and number of subsections need to be completed before the finale unlocks. The finale is always an escape sequence where the player must reach an exit before a new time limit expires.

Subsections can be used across a level to allow a player to learn a new mechanic by introducing additional complexity and interaction between other mechanics as they progress through a level meaning that in general, earlier sections will be easier.

If a player dies during a section, they their current section will be restarted, though the level timer will not reset, meaning timed doors will still close at a set time after the level was began regardless.

Pacing

As with many skill-based games, the game (after a tutorial) will open with an immediate feeling and look of high difficulty – although earlier levels should still feel considerably easier once the player has more abilities. This is to set an early precedence and allow skilled players to shine, similar how the Elden Ring pits the player against a mid-game boss as the first enemy they encounter – and those with an affinity for the Dark Souls series were still able to beat it.

The overall difficulty curve of the game will be moderate apart from the initial steep increase – but incredibly varied. This is due to how the player will gain more abilities not only as they reach further levels, but also as they explore and gain access to different exits to previous levels that might contain more abilities. This therefore means that if the player finds the difficulty overwhelming, they are encouraged to try one of the many secret route challenges in previous levels. Similar to open world games, the player may be faced with around 4 or 5 different challenge options at a time, ranging from a difficult level later in the game, or a set of different challenge doors. The adaptability of the difficulty curve comes from the fact that beating any one of the challenges will likely reward them with the ability and/or raw skill that will make most if not all of the rest of their current challenge set easier. The learning curve is intended to be long due to the number of ability interactions scaling with the number of abilities the player already has. For example, given n abilities, finding a new ability means that at most n new interactions can be experimented with.

In short, players will almost always have a choice between multiple challenges and will methodically be able to work through them one by one in a variable order.

Prototype Context

The submitted greybox prototype would be appropriate to use for level 8 if applied to figure 1 – though it would certainly be towards the start of the game where levels are more forgiving and therefore offer more of an opportunity for the player to become familiar with the core mechanics they had discovered thus far.

Notice that grenades are not a requirement in order for the player to beat the prototype level, and yet the example prototype player has already beaten level 7 and therefore gained the ice grenade ability (the also have the bouncing grenade which is unlockable in subsequent area, external from the given figure). This supports the aforementioned concept that completing challenges will often give rewards that in turn make other challenges easier, since players stuck on the prototype level (level 8) are encouraged to return once they have explored more and gained more abilities that would make it easier for them – the grenade in this case. This following section begins by highlighting a consequence of this pacing choice found in the prototype

Prototype Level Design Motivations

As demonstrated in the provided gameplay video and figure 2, there is a section near the beginning of the level that prompts the player to use an ice grenade (if they have the ability unlocked) on a group of enemies that are close together, which is optimal for such an ability. Choosing (or being forced) to not use the grenade instead makes this part of the level much harder due to the tight corridor and 4 sets of bullets and therefore mirrors the above statement about variable level difficulty.

Another section designed around game mechanics is shown in figure 3, where the player can chain dashes to traverse up the spiral of enemy pillars and reach a higher platform. While this section would be possible through jumping and dashing onto the platforms akin to a basic 3D platformer, it would be considerably slower which is undesirable since this part of the level is within the timed "escape" section. However, this has another layer to it – the timed challenge door just ahead. To access this section, the player ascends an elevator shaft of sorts using a fan. If the player manages to angle themselves correctly, they can use the fan's momentum to fly straight to the top thereby skipping the need to chain dash up the enemy tower and further allowing more advanced players going for the timer door challenge to more likely reach the door before it closes. Naturally, players are much more likely to use the simple route before the advanced route, which moreover results in this section serving different purposes for the same player once they replay the section with greater skill and experience in the game. Another consequence of this design is that returning players using a new save file (who have little in-game progression but high experience) can just as easily use this route to therefore skip levels they would find too easy despite not having many abilities – thereby further promoting the game's replayability aspect.

Finally, Figure 4 shows an area of the level where the player is intended to play out a series of actions (as described in the previous 'Dynamics' section), though it is expectedly more simplistic than later levels due to being in the first third of the game. A perfect execution would be to "pull bow, dash, fire, turn left, pull bow, dash, fire, turn right, dash and slide". A later sequence would use more mechanics at once.

Playtesting

Initial Testing Plan

I plan to run tests with some friends who are not on the games course but are familiar with difficult video games and standard controls, which is therefore reflective of the target audience and taxonomies but remain diverse within such communities.

For the greybox prototype tests will begin with acceptance testing of the core abilities, their usage when intended as well as intuitiveness of their interactions with minimal prompting. Then, the game will undergo exploratory testing to iron out any game-breaking bugs not found during initial prototyping in the editor.

I will then move onto a modified form of usability testing, where what needs to be communicated to the player will be tested using the limited greybox (non-artistic and mostly white text) UI. For after the prototype, additional usability testing would be required given the additional context of an in-game tutorial and more stylised UI/presentation.

Note that in the end, the second and third stages were swapped due to bugs being most relevant in the most recent version of the codebase. Further, while the iterations have been pre-categorised, some changes listed may fall between definitions such as between acceptance and usability, and thus the categories are general rather than rigid.

Iteration 1- Acceptance Testing

The first version only had the first section of the level finished, though this contained some abilities. Testers immediately agreed that the camera's vertical FOV needed to increase in order to increase the user's spacial awareness – which I, as the developer, didn't experience as much due to having built the space. In the full game this would be a graphical user setting in an options menu.

Against my own expectation, some testers did not intuitively dash into their own grenades as their intuition was that they would be damaged by a subsequent explosion, thus prompting the removal of the player's self-damage through grenade explosions.

Playtesting showed that, once testers became accustomed to the game's mechanics, the bpm of the level's song was highly correlated to the level's difficulty. I experimented with swapping out songs at 137, 115 and 105 beats per minute and found 115 to be a good personal middle ground which maintained a desired level of advanced thinking and reaction time. This would later be moved to its own feature as it opened the door for levels to be replayed with different songs at faster rhythms to increase difficulty without requiring developers to create much new content.

The gravity grenade had mixed opinions at this point. It challenged players to multitask in order to activate its strong gravity and drop it to the floor near an enemy at the correct time, though this proved more difficult than it was worth since testers with little experience would need to watch the grenade move forward to time their presses – which meant they were otherwise stood watching the grenade instead of the usual 'multitasking' emergent dynamic. Therefore, for this iteration, I added an indicator underneath the grenade that would show the player which object was directly underneath the gravity grenade.

The high difficulty of shooting grenades mid-air was immediately apparent, although this began to border on tedious for testers on lower-end hardware, especially since added input delay also changes the timing of their key presses as well as their aiming accuracy. To combat this, without any visual change, I added a larger hitbox to grenades that specifically only collide with the player's bow raycasting, which made shooting near the grenade still count as a valid hit and explode them early. This therefore made the grenade more viable for all players and increasingly usable at greater distances, while maintaining a high skill requirement and ceiling.

An example of a buffed mechanic would be enemy fire rates. This iteration only contained the yellow enemies that shoot once every 4 beats which playtesters eventually found underwhelming through repeated playthroughs once they gained a greater understanding of the game and its mechanics. This prompted two changes: the next, pink, moving enemies would have increased fire rates (firing on all beats) but would appear later in the level, and furthermore, the number of yellow enemies in the first section were increased thereby increasing difficulty due to more bullet hazards being in play at once.

Iteration 2 – Usability Testing

Despite the additional help with the downward indicator, the resultant strategy for the gravity grenade remained to stand and watch it rather than continue with gameplay in parallel and thus it was removed for the rest of this prototype.

The pink rapid-fire enemies would fire on all beats instead of every 4 beats like their yellow counterparts. Since dashing can only be done on beats, these faster-firing enemies would be difficult to eliminate sometimes as they would often shoot a bullet right as the player dashed, causing the player to dash into the instantaneous, sometimes not even visible bullet rather than the enemy. For this iteration, I therefore added the ability for the player to also have "dashed" into all enemies that are very close (technically, within a small sphere) to a bullet they dash into, thus eliminating both the enemy and their new bullet, causing these foes to feel fairer.

At this stage of development, dashes had a 4-beat long cooldown, but many playtesters recommended that it instead be refreshed upon eliminating enemies or touching the ground instead. This was because the cooldown's length caused two problems of which there was little middle ground. A shorter cooldown would have allowed the player infinite flight, as gravity was not strong enough to pull the player down enough and thusly, they would, over time, be able to move upwards in the air indefinitely. On the other hand, if the dash cooldown was too long, it would disallow the chaining of dashes thereby, in popular opinion, making the game less satisfying to play – and therefore the mentioned changes were made for the next and final iteration. This is an application of the MDA framework allowing playtest aesthetics to inform design mechanics.

Most testers experienced difficulty with keeping in time with the music since the placeholder songs used had not been heard before – which would also be true for a full game with its own original music – this was uncaught since I had heard the music many times through playtesting. Thus, I added UI to the player's reticle which helped to indicate when beats would be timed by the game, which improved player performance a lot. Note that this demo does not contain a method to calibrate the UI and music timing to systems with input and/or audio delay, but the full game should.

Iteration 3 - Exploratory Testing

Exploring for bugs was delayed as long as possible, since earlier code often became obsolete and new bugs appeared, and thus it was prioritised as late possible since the codebase would be more relevant to the coursework prototype submission.

Testers were asked to intentionally break gameplay, with most bugs and exploits being changed for the submitted prototype. An example was that, if the player intentionally waited in the level for the music to loop, the beat timing would be off – which was uncaught since the level was shorter than the song during normal testing. Another example would be that, if the player backtracked to previous checkpoints, their level progress would be reduced as well, which is unintended behaviour – leading to a check in the checkpoint script that would only activate checkpoints if the player had not already reached that checkpoint or later one.

Near the end of development, the player's movement was modified to use rigidbody physics rather than a character controller, which caused some changes. One bug was found during playtesting was that the player can "stick" to walls by directionally inputting toward the wall when airborne. This was actually kept in as a 'feature' since it added to the player's "weightless" feeling.

Figures

Figure 1 - Tentative level structure

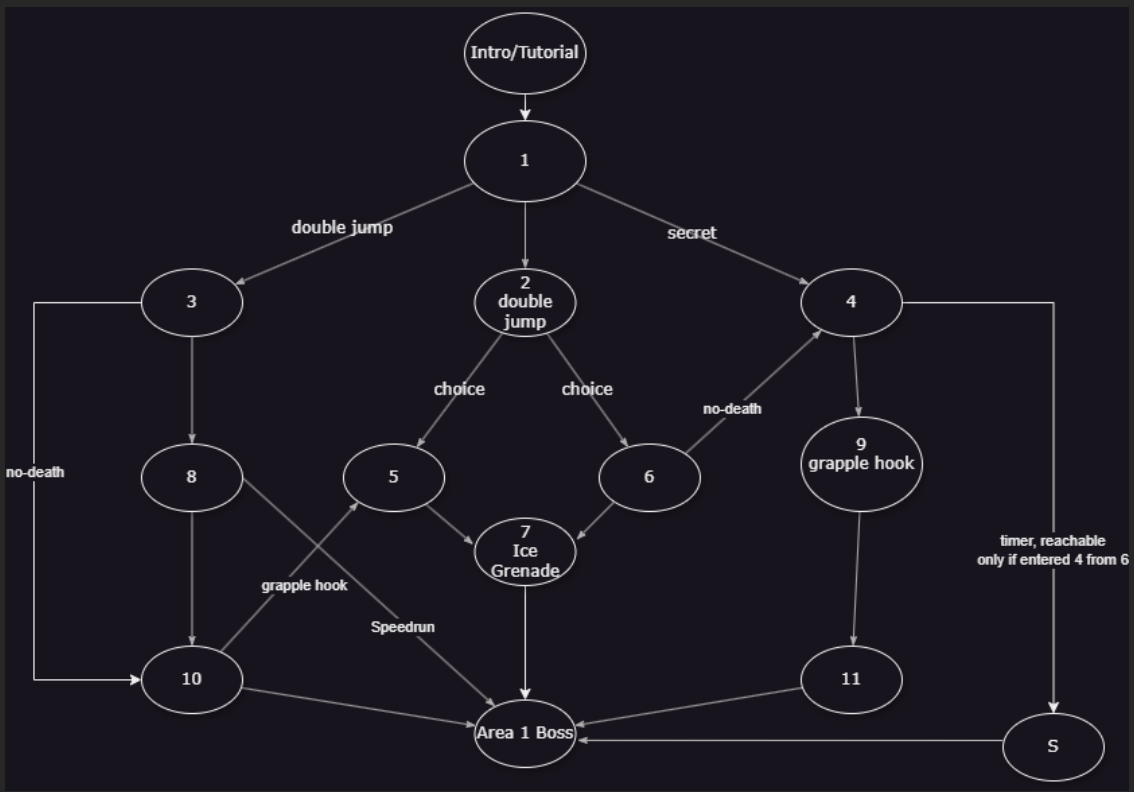

Figure 2 – Area prompting grenade use

Figure 3 – area prompting the use of dash impulse

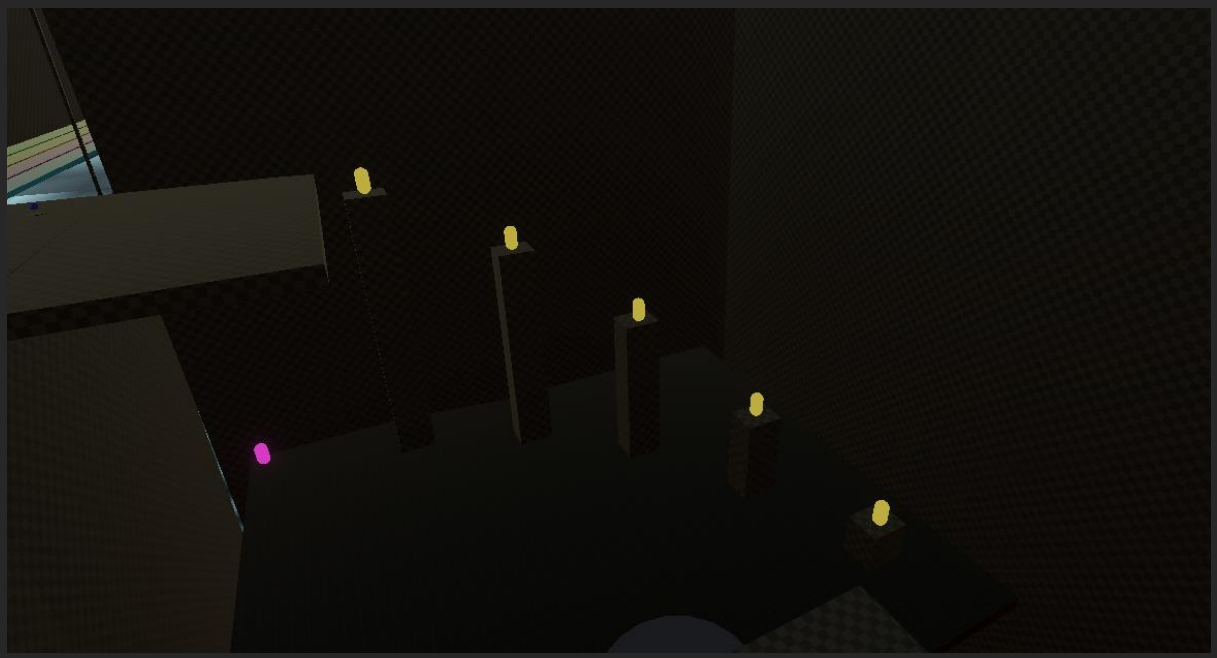

Figure 4 – Beginning of an area prompting the player to multitask their bow and dash abilities.