**Controlling the Player Character**

Flowing, precise movement is vital in a platformer game, and even more so in a speed-based game. It is vital that movement feels natural to the player, and success and failures of the past can show this.

**Bubsy 3D**

Bubsy 3D is the exact example of 3D movement done badly. The player only moves in the direction they are facing and is slowly turned using the control stick. The lack of movement in an axis separate from the camera causes the game to feel sluggish, despite the player not having momentum and stopping instantly when the button is released, and time is wasted by the player slowly rotating a bobcat to reach the next platform.

**What can we learn from this?**

Multi-directional movement is key. The player’s actions should be able to flow into each other naturally, without having to stop to make any key changes. Dead time walking around is to be avoided, as is control design that promotes frustrating interaction-based difficulty, as opposed to difficulty stemming from level design.

**Super Mario Odyssey**

Mario Odyssey has very fluid 3D movement. Mario has freedom of movement in all directions, and his multiple special moves are available at one button press, where mistakes made feel like the player’s fault, and not the result of bad design choices.

**What can we learn from this?**

Having special moves that do not affect movement, or effect movement in a very minor way, is important. The small momentum reduction when Mario throws his cap is a great example of this. It allows other movement options, such as the dive, to combo out of it, and allows for quick movement adjustment as a correction tool. This can be used in the development of aerial attacks in our game, stalling a player’s fall slightly to allow correction and other moves, such as a hook shot attack. Other movement options, such as the double and triple jump, are key examples of how combo-based movement is satisfying for the player. It is important that we develop a movement system that allows for different combinations of moves without feeling sluggish.

**How will this be done?**

The player will have momentum. Moving quickly in one direction only to turn in the other direction can be jarring if instant and feels unnatural to the player. The player will have influence over the player’s momentum through the controls, but the influence should be limited to ensure that movement feels natural. Jumping is a sharp increase in momentum upwards, and the player’s jumping arc comes from the gradual decrease in that momentum due to gravity, before falling in the opposite direction. Momentum can also be lost by not inputting anything, in which case it will be lost over time.

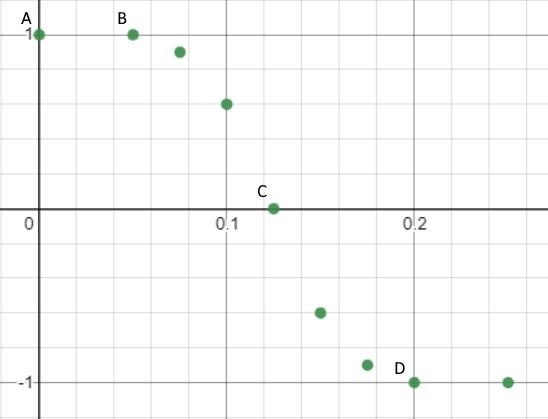
****

Figure - **Velocity** over **Time**

**A** – Player holding right and character moving at top speed.

**B** – Player releases right and begins to hold left, Character begins to slow down.  
  
**C** – Characters motion and ceased and they begin to change direction  
  
**D** – Character is now moving at top speed in direction opposite from where they started.

**Aerial Movement**

In the air, the player's effect on their momentum will be reduced, as turning around in the air is unrealistic movement. The effect will be roughly halved (subject to testing) but simply jumping is a commitment to movement. This adds tension to platforming and provides a more challenging experience to the player. All directional momentum will be handled separately, as is true in real life, so that moving upwards or downwards will have no effect on the horizontal momentum.

Other options will be available to assist the player, however. A double jump will be available, which can provide a speed boost in the required direction if needed, as well as a glide effect for large gaps. For more precise adjustment, attacking in the air will cancel vertical momentum, allowing a potentially failed jump to be adjusted slightly at the cost of some speed.

**Sprint button/Slowdown option**

A sprint button will be used to travel even faster. The player's momentum gain based on input will be increased while sprinting, as will their maximum speed. This, while allowing for better completion times of levels, causes turning around to be more difficult (see above), and if the player encounters an enemy, less time will be available to react. In theory, a player with a lot of level knowledge and fast reaction times should never have to stop sprinting through a level. This will be important to consider in our level design.

**Terminal Velocity, max speeds for objects and from objects**

The player will have set maximum speeds for travelling normally, without using abilities. However, abilities will influence this which must be considered. Using the player's shield slide, or the zip line, will likely cause acceleration beyond that which the player can normally reach. This is handled by not allowing the user's input to add momentum in this state. While travelling at a higher speed than their cap, the player's input in that direction will not affect their momentum, unless they are trying to slow down. The increased speed will last until the player slows down, or over time while the momentum is reduced. High-skill players can use this without slowing down for a bonus speed boost.

Vertical momentum will have its own speed cap, so that the player does not fall too fast if they hit a large drop. While this should never occur, having the system in place is important to prevent unforeseen behaviour

**Enemy collision and effects on speed**

Colliding with an enemy will instantly cancel the player's momentum, knocking them backwards. Different enemies will have different effects on this, but the player will be inconvenienced by having to speed up again after taking damage. The player will be able to knock enemies out the way with their weapons should they time it correctly, negating the speed loss and allowing them to continue.