Escape From Time Tower

An exploration of time manipulation mechanics in a gameplay environment with Unity

MComp

Computer Games Development

May 2019

University of South Wales

By

Jack Hodge

Jake Passmore

Liam Vallance

Nicky Jones

Oskar Dubas

Primary Supervisor: Dr Mike Reddy

Secondary Supervisor: Dr Carl Jones

Table of Contents

1. Abstract **3**

2. Introduction **3**

2a. Overview 3

2b. Aims of the Project 3

3. Literature Review **4**

3a. Time Mechanics In Games 4

3a-1. SuperHot 4

3a-2. Prince of Persia 5

3a-3. Braid 6

3a-4. Quantum Break 7

3a-5. Titanfall 2 8

4. Production of Game **9**

4a. Short business plan 9

4b. Specification 10

4c. Software Specification 11

4d. Contingency plans 11

4e. Assets 12

4e-1. Costs 12

4e-2. Free 12

4f. Development 12

4f-1. Player Character (Darren) 12

4f-2. Procedural Room Generation 13

4f-3. Time Mechanics 14

4f-4. Enemy Types 17

4f-5. Weapons Systems / Item Drops 18

4f-6. Bosses / Boss Rooms 20

4f-7. Models / Textures 22

4g. Testing and Feedback 23

4g-1 Itch.io Early Release 23

4g-2 Friends and associates 23

5. Conclusion and Future Work **24**

5a. Conclusion 24

5b. Future work 24

5b-1. Reverse Time Mechanic 24

5b-2. Time Travel Mechanic 25

5b-3. Procedural Room Generation 25

5b-4. Animations and Art Design 26

5b-5. Pick up Items 26

5b-6. Save/Load 26

5b-7. Audio Design 26

6. References **27**

7. Bibliography **28**

1. Abstract

This project uses the game created Escape from Time Tower in order to demonstrate the uses and implementation of different time manipulation mechanics into a game setting. This paper touches on previous games that have successfully used time based mechanics in their gameplay and discusses the the success of these mechanics and where things could go wrong if time manipulation mechanics are not properly implemented. The game will use each level to demonstrate a different time manipulation mechanic in order to show its effectiveness or ineffectiveness in the gameplay environment created. Discussed in the report will be how the mechanics may or may not work in this created test gameplay environment but also where any failed mechanics may be effective in other gameplay environments.

2. Introduction

2a. Overview

This project is aimed to explore and understand the utilization of time manipulation as a gameplay mechanic. Throughout this project we will be developing a isometric top down dungeon crawler shooter game as the main test environment and an example of these time manipulation mechanics. We will be manipulating time in many different ways separated via levels in the game world so that each time mechanic can be contained and utilized in a stand alone environment while maintaining a playable and enjoyable gaming experience. The methods of manipulating time in the unity engine will also be explored with a conclusion as to which method we found most reliable and why we feel that this method was the correct option to go with.

2b. Aims of the Project

1. To create a game with two different styles that do not usually go together (Gungeon style gameplay and time mechanical gameplay).
2. To research the different types of methods games and movies have used when dealing with time. Doing so will enable us to take the experiences from these games/movies to avoid mistakes or enhance what was great.

3. Literature Review

3a. Time Mechanics In Games

Time mechanics have been used in many different games of different genres over the years. This section will look into different games and how they have used a mechanic. Time mechanics are used in various different ways, in some games they are used as an ability for the player to use to manipulate objects and enemies in the game. An example of this is Bethesda's game Dishonored, where the player has access to an ability that allows them to slow down time for 12 seconds or more if the ability is upgraded, allowing them to sneak past or kill targets instantly.

3a-1. SuperHot

Time mechanics can be used as an essential part of the gameplay. Superhot, a game developed by Superhot Team, took advantage of using time as a key feature. In Superhot, time moves when the player moves, allowing every action that they take to be thought out. Superhot was also released to the virtual reality platform which took advantage of the player actually being able to move their body to unfreeze time.

The reason why this was a successful game is because the mechanic was simple yet very refined. Although time was based off your movement, it allowed them to further enhance the mechanic it self; forcing players to pick up objects while time is slowed down and throw them away to their advantage.

We used what was learnt from playing Superhot and applied a very similar time mechanic in our game, where time flow is based off the player's movement similar. Because of this, players have time to develop a tactic before proceeding through the first level.

3a-2. Prince of Persia

Prince of persia: The sands of Time uses a reverse time mechanic to allow the player to reverse the game back in time as long as they have enough “sand” collected to enable this. This mechanic was created to allow players to correct a mistake in battle or to try to traverse a difficult parkour section.

This use of a consumable and replenishable ability to rewind time is part of the key aspects of the ability in the gameplay design. It acts as a means to make the player work for the ability, rather than just being able to use it whenever they want by forcing the player to kill enemies and replenish the sand. It also gives the weapon, the dagger of time, an important role in both gameplay and story as it is the dagger that holds the sand that the player needs. The draining of sand from the enemies like blood gives a good reason for the enemies to disappear when they are killed, saving up resources (vital in for game development in the time of release) while also providing a story reason for this to happen (Mechner, 2008).

Time is an important aspect of not only the gameplay but the story of Prince of Persia: The Sands of Time. The whole story narrative uses this gameplay ability in a clever way. By having the game narrative told like a story where the whole storyline of the game is an actual story being told by the Prince as he recalls the past events of his history. This is done very cleverly as when the player fails by falling to his death for example the Prince will narrate over saying “No, no, no, That didn’t happen” implying that the action the player just failed was not part of the story the Prince is telling; as he obviously didn’t fall to his death if he is telling you this story. This as a gameplay element allows the player to fail and reset without feeling like it is a complete failure and more of a misstep that they will now correct on the next attempt.

A rewind mechanic is used in second level of our game in a similar way for a similar reason. It allows the player to make a mistake and take a hit but undo it, restoring their health (but also the health of their enemies) back to where it was beforehand. This mechanics seems to work well with the gameplay style here as the game is very difficult (even more so on the second level over the first) and so it is easy for a player to get overwhelmed by enemies. This way, they can use the rewind to get out of that tough situation and quickly rethink their approach.

3a-3. Braid

Platforming games have also taken advantages of using time as an ability to help them solve the puzzles, a game that uses this mechanic well is called Braid this was a game that had a different time mechanic for each of its different worlds, this had multiple time mechanics that the player could use to solve the puzzles including Rewind time, unaffected objects from anytime mechanics.

"If you think a game is '**Madden 2008**,' then hey, games probably aren't art."

“So with ‘**Prince of Persia**’ and ‘**Blinx**,’ they had this idea that it would be cool to have rewind in the game. But they didn't want to deviate from their core idea of what a game is about (you have a limited number of lives, and you fight guys and jump over traps, and if you get killed too many times it's game over). So in those games the rewind was just this sort of

superfluous thing. Yeah, it was kind of cool, but they didn't allow it to be really meaningful inside the game design. Because if they did that, it would have destroyed the rest of their design, removing all the consequence from the fighting or the puzzles.”

Two quotes by the creator of Braid Jonathan Blow, Blow had an inspiration to make a game that he felt was captivating, interesting and challenging focusing on the gameplay before the art design of the game, the first showcase of his game which won the [Independent Games Festival] game design award at the GDC in 2006 was filled with placeholder design showing that the award is for the quality of the gameplay and narrative over the look and feel of the game. Blow had issue with the way time had be utilised in games in the past as they did not have much of an impact of the games world or narrative and were too much of a gimmick for the players to play with (Totilo, 2007).

The success of this game shows the success that can be achieved with the correct outlook and use of time manipulation mechanics as it shows how thought into not only how the mechanics are used but consequences and rewards of using these mechanics can really play a big role in the feel of the game.

Had the success of the time mechanics system implemented in Braid not been so great it could have been at risk of being seen as this knock-off weird feeling Mario game with a painty design but instead it has stood out as a prime example of the ever growing indie development scene and great utilisation of time based mechanics in a gameplay setting.

3a-4. Quantum Break

According to Workman, (2016) Quantum Break is a Third-person science fiction action shooter that was developed by Remedy Entertainment. Players play as Jack Joyce, who has been given the ability to control time. These time powers allow players to use creative methods to defeat enemies with a combination of gunplay. Using these abilities the players will need to play a vast number of levels with a new puzzle or target to defeat with a combination of all the time abilities.

There are 6 different time abilities within Quantum Break :

- Time Stop
- Time Vision
- Time Dodge (Focus Time)
- Time Shield
- Time Blast
- Time Rush

All of these abilities can be upgraded to have longer and increased outputs.

Time Stop will freeze objects and enemies within a certain range of the affected area. This allows the player to use their weapons to kill these enemies. This also allows the player to freeze objects in time to allow them solve challenges and puzzles. (Workman, 2016).

Time Vision is the ability to be able to see the lay of the land. Time Vision shows objects within the surrounding area such as enemies weapons as well as items and collectables, this can be used when in combat to reveal enemy locations as well as useful interactables to help turn the fight within combat. (Workman, 2016).

Time Dodge gives the player the ability to evade danger for a few seconds, this can allow for the player to get behind enemies to attack from behind, it can also be used to dodge grenades and other attacks. (Workman, 2016).

Time Shield is the defensive ability that can protect the player from incoming enemy fire. It allows the player to regenerate their health and plan their next move to overcome the targets. It can also be used to shoot through to kill enemies while being protected from danger. (Workman, 2016).

Time Blast an offensive ability can be used to take out single or multiple enemies within an environment, this is the only time ability that can deal damage and kill targets by blasting them away. (Boccher, 2015)

Time Rush is the ability to allow the player to freeze time within the entire level and allows the player to dash to another location within a split second. Further in the game this ability can be used with other abilities to take down targets easier and faster. (Workman, 2016).

All of these abilities are used with a stamina bar so each ability can only be used so often within a given time (Workman, 2016).
Of these abilities Time Rush and Time Freeze will be most beneficial towards the development of Escape From Time Tower. A Pause Time Function has been used within the third level of the game, this Pause Time will work similarly to the Time Freeze function within Quantum Break. The ability will allow the user to pause time for a short period, freezing all enemies within the level, only allowing the player to be able to move. This ability will only be usable for a short time before the function time ends and all the enemies within the scene are moving once again.

3a-5. Titanfall 2

Occasionally Time Manipulation mechanics can be used not as a main mechanic in the game but as an alteration to the gameplay. One of the levels in Remedy's First-person perspective shooter Titanfall 2 (Remedy Entertainment, 2016) called "Effects and Cause" breaks the typical run and shoot gameplay of the game and introduces the ability to travel back in time, allowing the player to experience two versions of the same level. Player's ability to travel between two versions of the world is instantaneous and with no limitations. This time travel mechanic created a good opportunity to tell a story of the past events, introduced new puzzles and new ways to deal with enemies, everything without breaking the fast pace of the main gameplay. This level is considered one of the best moments in the Titanfall 2 by many (Hamilton, 2018) and it is a good example that Time Manipulation has a place in any type of game, and it can enhance the gameplay in many ways.

4. Production of Game

4a. Short business plan

The game will initially release as a free test build (beta) on the indie game site itch.io, this will give us the ability to have an early build of our game tested my many people from around the world providing us is real consumer feedback on the development of this game, with this feedback the build of the game can be progressed and pushed towards a final build which can be listed as a “pay what you want” title with our desire to at least reach the £76 publish fee required for us to publish to steam if the game proves to have some success.

The game will be released for PC using Steam - a digital distribution platform for video games developed by Valve Corporation if the release on itch.io proves successful. Console release (PS4/Xbox One/Switch) will be possible only if the game is successful on the Steam platform in order to justify the purchase of the console dev kit. Mobile platform required additional work to translate twin-stick mechanic into a touchscreen, which may require additional development time. For this reason, the mobile platform can be considered after potential success on Steam.

Steam Direct, Valve’s Developer Program to distribute a game, requires a fee of £76 to publish a game. Currently, we have no plans for microtransactions or DLC, therefore no additional costs are needed for the Steam release. However, if our game ever reaches a $100k revenue we will be required to purchase a Unity Plus account which costs $35 per month.

The Game upon release will have a price of £2.99 ($3.99). This puts the game in a price range way below high detailed pixel graphics indie games like Enter the Gungeon (£10.99) or Starward Rogue (£8.99) and above simple 2D browser Shoot Em Ups like Bit Blaster XL (£0.79) or Ink Plane (£0.79). Our price is in the range of simple indie Action games, usually with pixel graphics and simple chiptune soundtracks like Rush Rover (£3.99) or Skelly Selest (£3.99). There is a possibility of reducing the price during the Steam Sales to anything between -%70 and -%90.

4b. Specification

We'll be using time mechanics from games such as Superhot, Max Payne and Overwatch with a Gungeon style gameplay using the Unity game engine (2018.2.21f1).

The object of the game is to get "Darren", a daredevil but charismatic fellow, to the bottom of Time Tower before it explodes. Each level will be represented by a different time mechanic that can be used in-game to challenge the player while progressing through those level, the different time mechanics are as follows: Slow down, Speed up, Stop time & Reverse time.

Slow down time mechanic: The way the slow down time mechanic will be implemented is based off the player's speed, if the player continues to move at its maximum movement speed time will flow correctly, however each time the player slows down for whatever reason so will time.

Speed up time mechanic: The speed up time mechanic will be the only time mechanic that is based off player shooting, the player will shoot a time bubble bullet that when it hits an object it will create a bubble of time which when a player, npc or moving object enters will drastically increase their speed to simulate time.

Stop time mechanic: The stop time mechanic is as the name suggests the ability to stop the movement of everything (that is capable of moving) when the ability is used.

Reverse time mechanic: When the player reverse time it reverses time of everything that moves in the level, it is limited to a short period of time because the scene captures each frame and inserts it into an array which is how they are able to reverse time, so to prevent a massive performance hit the number of frames the array can hold is limited. So the player can not get to the end of the level and then reverse time back to the start of the level, the player will also be able to stop half way through the reversing time, if for example the maximum time that can be reverse is 5 seconds then the player can stop reversing time 2 seconds into that.

With some of the these time mechanics there is the possibility that they can be used in a way to make each level very easy if exploiting them correctly (with the exception of the slow down time mechanic as that is applied to the players movement and as such comes with its own risk/reward), so each time mechanic will have a cooldown usage applied to them to prevent this, but not only that it will add an extra level of thought in the player's mind to consider as the cooldown will potentially be lengthy and punishing, so although the abilities are powerful, if used incorrectly/inappropriately it could cause the player's death.

4c. Software Specification

Software that will be required for the project :

- Unity 2018.2.21f1
- Visual studio 2017
- Blender (if creating own 3d models)
- Inkscape (for 2D gun graphics)
- Github
- Google Docs

4d. Contingency plans

In the event of unforeseen circumstances a contingency plan is necessary to ensure the project stays on track. For example, some features we initially planned to include in the game could prove too difficult to implement or too bug-ridden to fix in time for deadlines. To combat consequences of this we need to focus on good object oriented programming techniques early on in development cycle to ensure each feature is as self-contained as possible. We can then drop any problem features from the final build with little to no collateral damage.

On the other end of the scale, planned work on the project could be completed earlier than expected due to either Unity engine's wide range of tutorials and resources or the inclusion of 3rd party libraries in our product. Although this will help us achieve a playable build quicker, it will also leave the team with little to implement to make our game unique. Therefore, it is also important to record a log of all ideas in the planning stage, even if they seem to be out-of-scope for the timeframe we have. This log could then later be used as a back catalogue of further features.

A potentially disastrous situation could arise in the event that a team member cannot perform their duties or leaves the project for any reason. In this case their responsibilities could prove vital to the state of the project and result in an increase in workload for all other team members. This is why it is important to assign smaller workloads to all members; simultaneously reducing the impact of a missing person while also providing everyone else with enough time to tackle additional jobs.

As with every software engineering project, data loss through hardware, software or network failures possess a very real danger to the survival of the work. Although these sudden faults are often unavoidable, precautions have been made the ensure no major data loss occurs. This includes using a version control system such as Git to keep a copy the work as well as

all history in the cloud, and each team member individually keeping their own up-to-date version of the project on their home computers.

4e. Assets

4e-1. Costs

Because the group felt that the default Capsule models that were created at the start of development for the enemies were not good enough, it was decided that the Unity Asset Store was the next best place for help. There was a very good model package for $5.36/£4.26 inc VAT, the name of the package is Fantasy Robot Pack 1, the reason this package was chosen was due to the fact it had both rifle and melee models with animations already created for both, this allowed them to be imported relatively easily and only had to create an animator for the animations and call them appropriately during code.

4e-2. Free

There were four texture assets used from the Unity Asset Store to create the levels realism, these assets are as follows: Free Sci-Fi Textures, eU Sci-Fi textures, Sci-Fi Texture Pack 2 and Futuristic Panel Texture. A Unit Frame asset was also used for the player's health to be shown on the UI called RPG Unitframes #1.

4f. Development

4f-1. Player Character (Darren)

The key most important and memorable part of a game is arguably the player character. The player character is a key tool to the narrative of a game supplying backstory, reason and controllability to the story. The player character is very commonly the main tool for interacting with the game environment allowing the user to maneuver around the game space and interact with objects around it. In Escape From Time Tower the player character is named Darren, having a name gives the player character an identity for the player. Darren is tied directly to the narrative of the game being an experiment in the the topmost highly secretive floor of the tower. After a major accident that resulted in an explosion in the third floor, the player finds themself freed from their cell and with the goal of survival and escape from the tower.
This narrative is what provides the player with the goal to control Darren though each room on each floor in order to fight their way to freedom.

The early designs of the player character were very basic pill shaped objects which simply provided the ability to build the player movement for maneuvering around the game space. The next design addition was a simple weapons function to the pill object that allowed the player to move and shoot, these were the base designs allowing for further development of the game test bed such as the scenes, basic time mechanics and some enemies.

Once the game was in further development the player character needed a bit of a personality upgrade, with the lack of any designers in this project team a very simple design was picked of just a red cube structure with two gun arms and a face, this face sparked the interest in a “cute” and “silly” styled character that could remain memorable while also providing possibility for an upgrade in the future through commissioned design work. The face of this character was also designed to change from a “smiley” face when not shooting to and “angry” face when shooting which added some charm to such a simple design and thus Darren was born.

Darren later went through another simple upgrade in his character design, **H.A.T.S**, as we know there are no such things as “bugs” and are simply “features” we did not intend. Well we had one of these “features” once the enemy models got an upgrade and due to the fact that the enemies shot straight forward and not down at Darren we discovered that due to scaling issues the ranged enemies now shot over Darren's head. This meant that the player could no longer take damage from enemy projectiles. Simple scaling changes could have solved this problem easily but an opportunity was spotted and so a new system was implemented we tactically called the **Highly Advanced Targeting System**. This system added changeable hats for the player character, these hats increased the height of the player character allowing for the player to take damage once again while also providing the player with some customizable personality for their character through the wardrobe scene, where a selection of hats are accessible.

4f-2. Procedural Room Generation

In order to give players a different experience each time they played this game, procedural generation was added to randomise the map layout each time a level was loaded. This technique is fairly common in small development teams since it vastly reduces (or outright removes) the large amount of time needed to manually create levels (Davis, 2017).

Through this system, the programmer can set variables such as how big/small the rooms will be, how many of these rooms the player will need to beat before reaching the boss and how frequently enemies will appear in these rooms. The procedural generation algorithm will then use these values to create a room with 1-4 doors in each direction before populating it with enemies. Once these enemies have been beaten, the doors will open and new rooms will spawn; allowing the player to make their own path through the level. Upon completion of the final room, the algorithm will then spawn a man-made boss room perfectly rotated and aligned with each possible exit door.

4f-3. Time Mechanics

In order to manipulate time on GameObjects, two fundamental classes were needed in order to control both global and local time dilation:

Unity Time Scale:

Unity Time Scale is a very effective variable that can be used to manipulate how time is perceived by all GameObjects in a scene. This affects everything from how frequently a script's Update function is called to how Unity's underlying physics engine acts. It does this by applying time scale to the delta time (time since last frame) variable each frame. For example, if a Time.timeScale of 0.5 (half speed) is set and it takes the system 0.03 seconds to produce each frame; any objects referencing Time.deltaTime will instead receive the value 0.016 (0.5 * 0.03). This will cause time dependent code such as timers or movement to think only half the amount of time has elapsed since the last frame.

Although Time Scale is very useful when slowing the entire scene down, there are some also some circumstances where this feature would not be useful. The first is in the case of a stop time mechanic. Here Time.timeScale would need to be set to 0 in order to freeze everything in its place and although this would be acceptable for things like pausing the game, problems would occur when allowing certain GameObjects to carry on unaffected. This is because since Unity's physics engine is tied into using delta time, it's gravity and collision detection would also be non-functional.

Another instance where a different technique would need to be considered is if only a relatively small amount of GameObjects would need to be affected at one time. Since Time.timeScale is a scene-wide modifier that affects built in Unity features such as physics and collisions, all new classes created throughout the remainder of the project would have to be programmed around this potential change.

Local Time Dilation:

The Local Time Dilation class was created to address the shortcomings of Unity's built in time dilation and acts as a middle-man between that and the time mechanics themselves. This class is attached to all GameObjects that can be affected by time modulation and has its own dilation variable to apply to delta time as opposed to using Time.timeScale. LocalTimeDilation.getDelta can then replace Time.deltaTime in any scripts attached to the object in order to obtain the locally manipulated delta time.

This is beneficial since it has no bearing on Unity's global physics and collision system, meaning that any object can be frozen through setting their own dilation to 0, yet it will still detect and act on a collision if another unfrozen object touches it. Stopping time was a major problem with Time.timeScale since any object that moves through Time.unscaledDeltaTime whilst the scale was set to 0 can move through solid geometry ignoring all collision and trigger events.

There currently exists 5 different time mechanics in the game; they are as follows:

Slow down time:
Slow down time was a fairly simple mechanic to add because of how easy Unity handles global time dilation. Here, the rate of time in which the game is playing is controlled directly by how fast the player is moving.

In order for us to make the slow down time mechanic to our specifications, we had to link Unity Time.timeScale variable to the player's movement. This was easy to map since all movement inputs come together to form a single direction vector in the Update phase of Unity's order of execution. The magnitude of this vector can then be obtained by the slow time mechanic to judge how far the player is pushing the stick or if they are pressing any movement keys, before applying it to Time.timeScale. To maintain the physics in our game we also had to adjust the variable that affects the interval in seconds in which physics and other fixed frame rates gets updated.

What allowed us to refine this feature more and introduce new abilities alongside it was the fact Unity also has an non-affected delta time variable called Time.unscaledDeltaTime. This runs as Time.deltaTime would, as if not being tampered by Time.timeScale. With this, we were able to create objects that aren't affected by the modulated time scale, such as the Time Gun which can shoot normal speed bullets regardless of if the time is slowed down by the player.

Speed up time:
The speed up time mechanic works similar to the slow down time mechanic in that it adjusts an entities speed. However, the speed up time mechanic isn't based off the players movement, but a local time bubble that is created by the player. The player will have the ability to shoot a time bubble bullet, which, when it hits an object will create a time bubble. The speed up mechanic is then applied to everything within the bubble, including the player, enemies and bullets, two different methods were discovered when creating the implementation of the speeding up effect.

Method 1:

> Originally, there was no way to adjust the timeScale for specific entities in Unity. Therefore, the first method used was applying a force to all affected entities' rigidbodies. This was achieved by gathering the direction vector of the GameObject, and pushing its rigid body further in that direction each frame. This gave the illusion as if time itself was sped up in the bubble.

Method 2:

> A second method was later implemented once the Local Time Dilation class had been created. This method involved adjusting the LocalTimeDilation.dilation variable for each GameObject that comes into contact with the time bubble and reverting it once they leave. The result of this meant that all time dependant code ran by the affected GameObject (including timers and animation controllers) were also affected, giving a more realistic look.

Reverse time:
The reverse time script does not utilise the Unity timescale at all and instead all the functionality is handled with the scripts created for it. The reverse time script that handles the recording of the gameObject data is written in a open ended way without direct reference to specific objects meaning that it works on any object it is attached to. The game objects that the script is attached to will then contain their own list of previous data called pointsInTime, this list is being updated with the objects positional data, rotational data and health value (only if the game object uses a health manager), the list contains 3 seconds worth (we decided 3 seconds for balance and performance reasons) of these data points popping the oldest off the bottom when the list is full to make room for the newest object data.

Once the player activates the reverse time ability, every game object which contains the reverse time script steps backwards through their list of of previous data points setting their current values to equal the listed set each step, these object step back through the list until the lists are empty at which point the game resumes as normal and every reversible object begins filling their lists again.

During the rewinding process any object that uses a weapon such as Player, Enemies and turrets are temporarily prevented from being able to fire their weapons. Without this, enemies while rewinding would continue to shoot at the player character breaking the illusion of time flowing backwards as well as leaving the player in a defenseless situation causing frustration at the game.

While rewinding if any object reaches their point of creation (such as a the moment a bullet was fired from a gun) the object will be destroyed to help give further the illusion of time flowing in reverse as if time continues to move backwards then that object did not exist at that point in time and so should not persist as time resumes as normal.

Stop time:
The stop times freezes time within a level. When time has frozen everything but the player is frozen, enemies are unable to move the same for bullets that are active within the the scene.

Method 1:

> Time stop was created by using Time.timeScale, this was created by using time.unscaledDeltaTime on the player and ScaledDeltaTime on everything else. By doing this it will allow the player to move when the time.timeScale is equal to 0.

Method 2:

> Just like with the speed up time mechanic, a second method was later developed that uses Local Time Dilation in order to fix problems created by the use of Time.timeScale. In this method, when time is stopped the script iterates through all enemies and bullets currently active in the scene and sets their LocalTimeDilation.dilation variable to zero.

Time travel:
This mechanic was implemented later in the development. It allows the player to move instantly between two versions of the same level. In one, the player will encounter enemies as usual (present time) however, in the other one (future), there won't be any enemies, but the layout of the level will be designed specifically to block player's way.

In this technique two copies of the same level are placed in a specific distance from each other in the same scene and player model will move on the x-axis between them. The camera also needs to be moved on the same axis at the same time. Otherwise, the transition won't be instantaneous. Unity engine can correct the position of the player automatically if he accidentally teleports into a wall that is present only in one of the "worlds".

In this level, the quality of the gameplay is partially depended on the quality of the room design. In order words, the travelling between two different worlds only makes sense if the level itself is built around this mechanic. For that reason, levels must be created manually. This approach prevents the usage of the procedural room generator. An appropriate method needed to be created which can accomplish necessary processes presented in room generator function in order to start a level (e.g. spawning player and camera, setting the enemies and powerup drops).
Enemies in the present world are spawned randomly using the random spawn algorithm. They are created in the area further away from the player spawn point, preventing the player from being swarmed with enemies from the start. In the "future" level obstacles were meant to be spawned randomly around the level using the same algorithm as enemies. Unfortunately, during the testing, this approach caused too many glitches and inconsistencies in level difficulty, so now walls are placed manually.

4f-4. Enemy Types

Three different common enemy types were developed for Escape From Time Tower. They are as follows:

Melee:
Melee enemies are the simplest enemy types and simply switches between wandering around searching for the player, and chasing it when a line of sight has been established. The enemy can break this line of sight by quickly moving out of the bot's field of view either through the use of dashing or time mechanics.

Melee enemies deal damage instantly upon colliding with the player and then periodically every second until the player moves out of range.

Ranged:
Ranged enemies begin with the same basic wander behaviour as the melee enemy, but instead use ranged attacks when they eventually find the player. On spawn, these enemies are equipped with the same basic handgun that the player uses, but like the player, they can also make use of weapon drops from fallen enemies.

If the player is then spotted at any time, the AI will switch into its fight behaviour, making use of the predictive aiming (aiming at where the player is going to be, as opposed to where they currently are) to shoot the player. This behaviour still makes use of the basic chase action, but will also stop short when in an acceptable distance since there is no longer a need to get close.

An additional behaviour has also been added to the ranged enemy to allow it to actively seek out and collect gun power-ups, before switching weapon to the more powerful option. This behaviour holds lower priority than the fight behaviour and will therefore only be activated if no player is detected.

Hybrid:
A common hybrid was added later into development to give the illusion that the melee enemy AI was much more dynamic than mindless drones. The hybrid enemy runs off an external state machine script to allow the enemy to switch seamlessly between the previously mutually exclusive melee and ranged scripts.

To achieve this, the melee enemy was given the option to be able to search and collect power-ups similarly to gun enemy. When the hybrid script detects that one of these has been picked up, it shifts the enemy onto the ranged AI to allow it to aim and fire its weapon.

Since unlike the ranged enemy, the hybrid enemy initially spawns with a melee AI; the enemy itself does not possess the infinite ammo pistol. This means that once the ammunition for all guns picked up by the hybrid enemy is depleted, the AI reverts back to the melee script.

4f-5. Weapons Systems / Item Drops

Weapons
To add some variation into the ‘run and gun’ playstyle of our game, multiple different weapon types have been added for use by the both the player and enemies. Originally, all advanced weapons were treated as power-ups for the starting handgun and therefore ran off a timer which allowed the user to fire the gun as many times as they could before they lose it. However, after a playtesting this system, it was decided that they would instead have limited ammunition. This removes the pressure of forcing the player to use their newly acquired weapon before the timer runs out and allows them to save their ammunition for more pressing situations. Because of this, a weapon switching system also had to be implemented into the game. This makes use of the number keys and scroll wheel when using PC controls and the left and right bumpers when using a controller.

Handgun:

The player and all common ranged enemies spawn with a basic starting handgun. This has infinite ammunition and is therefore accessible to its users immediately. The low damage inflicted by this weapon encourages the player to seek out more powerful weaponry from fallen enemies.

Shotgun:

The shotgun was originally implemented by creating a new pellet prefab containing five of the normal bullets each facing a different direction. This would cause them to spread out in different directions when instantiated to mimic the effect of a shotgun spray. However, this was later changed to instead instantiate multiple regular bullets and change their direction vectors at runtime. This allows us to tweak and randomise the degree of the spread easily.

Machine Gun:

Since all guns run off the same abstract base class, the machine gun was created by simply increasing the rate of fire of a regular gun.

Extra-Damage Handgun:

The extra-damage handgun was created by increasing the damage of the regular handgun. The fire-rate of this weapon was also lowered to make the player consider the trade-off of using it in each situation. Because of this, this gun is better suited to more accurate players.

Time Gun:

The time gun works just like the regular handgun but fires bullets that aren't affected by any time dilations. This was created through the use of a new bullet prefab which uses unscaledDeltaTime as its method for moving, this prevents it from being affected by time dilations. Use of a different prefab not only made the implementation of this gun easier, but allowed the look of the bullet to be customised in order to stand out from the rest.

Weapon Drops

Within each of the levels, the player has a chance for a weapon drop, these drops are split to a random generated chance of them being dropped by an enemy when they are defeated. These weapon drops include a machine gun, shotgun and a weapon that outputs extra damage.

When these power ups were first implemented they originally used a timer, to determine how long the player could use it for once they picked it up. In the early stages of the development of the power ups the player could only collect one power up at a time and if they picked up another power up it would override the existing power up that they had equipped.

Health Drops
After realising the game may be a bit to difficult for the novice/casual player we decided to make some changes. One of these changes were the addition of health drops. In order to reduce the chance of useless drops, these only drop if the player has lost health. The lower their health is to zero the higher the chance of health packs being dropped from the enemies.

To keep the balance of the game in check it was decided that if the roll should fail and a health pack is not dropped from the enemy then it will drop ammo for the player instead, originally if the player failed its roll that would be it that no ammo would drop and that would be the punishment for losing health, but it was felt that the player shouldn't be punished if they have lost health by giving them any ammo to kill more enemies for a higher chance of a health pack dropping.

This allowed us to tone down the difficulty drastically while maintaining the motivation to not get hit but if they did then there is still ammo drops to allow them to kill the enemies easier/faster.

4f-6. Bosses / Boss Rooms

To allow for maximum compatibility with early mechanics, each boss AI is built upon a foundational abstract enemy class. By default, this allows the bosses to access all common behaviours such as chasing and wandering. As well as this, each boss has been given their own specialised behaviours and level room in order to make them stand out from the common enemy.

These bosses are as follows:

Jimmy the failure:

This is the first boss you'll encounter in the game, as it is an introduction boss to the player the general playstyle of the boss is not dissimilar to that of the normal enemies the player have encountered so far, The boss has a considerable amount of health compared to the enemies and have three unique mechanics that isn't shared by the enemies.
Gun Switching: The first mechanic the player will be introduced too is the gun switching mechanic, as the name suggests at different health percentages the boss will which the gun it is currently using to a different one to add more spice to the fight, currently the boss changes from a normal pistol gun to a shotgun and then finally when the boss is less than 15% health it will switch to a powered up pistol which deals more damage than any other gun and will be the most crucial part of the fight for the player.

Teleporting: The second mechanic the player will be introduced too in the middle of the fight is teleporting, when the boss reaches half health it will randomly teleport at a different location inside the room.

Turret: The third and final mechanic introduced into this boss is a turret inside the room, in the centre of the room there is a turret pillar that will periodically shoot different types of guns out from pistol to shotgun and in different positions and patterns to keep the player on their feet and always being vigilant about their location.

The Insane Missile Joe:

The main feature of this boss level is the Boss homing missile that it fires at the player instead of the standard gun. Once the player has entered the room the boss will not engage them until they have entered the main circle of the arena, this being the circle of white walls that the player can use as cover to avoid taking damage from the missiles. Once the player gets behind the pillar then the boss will lose sight of the player and will stop shooting. This gives the player time to think of a new angle of attack to defeat the boss.

King Danny-Rhys:

The stand-out feature of this boss is its ability to dual wield ranged weapons (specifically machine guns). Upon spawn, the boss is placed at the centre of the room, facing the door and remains rooted to this location throughout the fight. When the player enters through this door the boss starts rapidly firing its weapons aimed slightly to the left and right of the player, trapping them in the centre of its view. From here, the boss runs off its first behaviour of simply swaying its guns left and right for ten seconds, forcing the player to comply with its movements to remain unharmed. During this behaviour, the player has the choice to stay far away from the boss where the large distance between bullet streams give them comfortable room to move around, or move in closer to obtain a better attacking position.

After these ten seconds are up, the boss begins its real attack of angling its guns in at each other, making damage unavoidable should the player stay between the bullet streams. This forces the player to either stop time and step out of the boss's centre field of vision or dash through the bullets and potentially take some damage. This creates the prime opportunity to deal damage to the boss from its side.

Once the behaviour is finished and the boss realises that the player is no longer in front of them, it will maneuver its guns to their widest angle and turn toward the player with only the trailing gun (relative to the player) firing. This lulls the player into a false sense of security since if they want to keep rotating around the boss to hit it from the side they simply need to outrun the furthest bullet stream. However, once the fore gun (relative to the player) reaches a position where it can fire ahead of the player from the angle they are running at, it turns back on to trap them once again.

David Sawridge:

Due to the irregularity among the number of time mechanics (4) and bosses (3) towards the closing months of this project, a fourth and final boss was created to make use of the speed-up time feature. Because of this, the playstyle needed to tackle this boss were designed entirely around this mechanic. Upon entering this

room the player will notice the boss's saw blade begin to spin and its colour turn from its original metallic red to green. Once blade is at a threatening speed, the boss will begin to chase the player around the room in an attempt to cause melee damage; prompting the player to keep their distance and shoot back. Unlike the other bosses, bullets will do much less damage against this enemy and will cease to do any damage at all once this chasing phase is over, encouraging the player to find another tactic.

After ten seconds, the boss's saw blade will stop spinning and the boss will return to its starting position to ready itself for the next phase. Once here, the boss will start to slowly spin its saw again and change its colour back to the original red as it charges up its next attack. This is the signal that bullets will no longer do any damage. Once the colour is fully red, a devastating lunge attack will occur. Here, the boss will calculate the distance from its position to the wall (through the player), and from this, deduce how long it will take it to cover this distance and stop just short of the wall at lunging speed. The player can combat this however, by using the speed-up bubble on the boss before dodging out of the way, causing it to overshoot its calculated run and slam into the wall.

Once this attack is complete, the boss's red colour will drain back to green to begin chasing the player again.

David Sawridge Boss Room:

The David Sawridge boss room plays a part in how the player can defeat David Sawridge, as mentioned if the player is able to use the speed-up bubble and force the boss to slam into the wall this will cause damage to David Sawridge.

This tactic won't be shared with the player ahead of time but designed for people who will take the time to study the boss and the possibilities involving the speed-up bubble.

4f-7. Models / Textures

After a couple of months of development, the decision was made to replace the placeholder models created at the start for something more substantial. It was decided the best course of action was to purchase an asset on the Unity Asset Store which contained a number of models with animations. Alongside this, a number of texture assets were also used to help improve the level design. Allowing players to feel more immersed in the world.

4g. Testing and Feedback

4g-1 Itch.io Early Release

For external testing and reviewing, the indie development site itch.io was used to post a free-to-download alpha/beta version of the game with the intent of receiving real customer feedback on the game design, structure and balance, as well as bug reports. The release was uploaded to itch.io on the 22nd of January 2019 and promoted on the Escape From Time Tower FaceBook page. This version then had a hotpatch on the 23rd which added the bottom wall fadeout as it became apparent very soon after the itch.io push that this greatly hindered the player experience when playing near the bottom wall. As expected with this being an early build version of the game, this alpha did not receive much attention with only 52 views on the game page and 10 downloads. However, it did serve to get the game name out there on the site and worked as a test for pushing a build of the game to itch.io ready for the next more advanced build.

4g-2 Friends and associates

The majority of game feedback has come from friends and associates who were asked in person to trial varying builds of the game and provide verbal feedback to a member of the project team in which this feedback was passed on to the rest of the group, discussed and action taken based on the outcome of these discussions.

During the early weeks of April 2019 team member Liam Vallance spent a week working at Future University in Hakodate Japan collaborating with a number of students on their projects. In their spare time, a number of the Japanese students were happy to trial a build of the game and provide their feedback of what they liked and felt needed adjusted. The general reception of the game was well received and all the students agreed the fast pace of the bullet hell game style was enjoyable and challenging and the different time mechanics per level added a new interesting dynamic. The main feedback given was that the game was a little too difficult and this could hinder the enjoyment of the game, as enemies took a lot of shots to kill while the player only took a few and at this time there was no form of healing mechanic in the game. This feedback was discussed by the group and it was based on this development of a health pickup system to be implemented into the current ammo drop that the damage and health values were adjusted accordingly as well as sparked the system.

Throughout development team member Nicky Jones has regularly showed the progress of the game to multiple friends who specialise in playing shooting games of all genres, this has helped guide the development of the project. Many of the frequent positives is the fast past and constant action the game provides which a lot of people seem to gravitate to when first playing the game, the more common complaints however have been the graphic style of the game being very poor and mediocre. Other common complaints included the general quirks the game offers like animations not working quick enough, random bugs and the way the AI can lose focus so quickly.

5. Conclusion and Future Work

5a. Conclusion

In conclusion this project was a success, the general goal of the project was to investigate the use of time manipulation mechanics in a gameplay setting. In order to test this we decided to put these mechanics into a well established genre that would not typically mix well with time manipulation mechanics. The challenge we set ourselves was to research into previous games that successfully utilised time manipulation in both their gameplay and narrative aspects. We took these mechanics and tried to make them work well with the bullet hell genre by making each level of the game use a specific time manipulation mechanic as the key to the success of the level and defeating the boss.

This project provided the opportunity for us to experience a like industry team management system in which we used GitHub source control to keep the project flow with multiple members. Each member was assigned tasks and great communication throughout the project between the team members was the key to the success of the project.

5b. Future work

5b-1. Reverse Time Mechanic

There were some aspects of the reverse time mechanic that could have been improved or expanded on with more time to work on them. There are two main reasons for certain aspects not being included, either the feature would have been too complex to implement within the timeframe of the project or the feature was so trivial and unnecessary for the functionality that it was left out so that time could be put into more urgent and important tasks.

One of the biggest advancements to the reverse time mechanic that could be implemented is the handling of dead enemies and destroyed objects. In the current state if an enemy dies or a bullet is destroyed then they are gone for good, even if the player rewinds the game to a point before their death or destruction they will not return. With more time to expand this feature objects could simply be disabled instead of being fully destroyed when they die or are destroyed and remain for 3 seconds before they are finally deleted from the game world to free up resources, this way they could continue to record their data and upon rewinding could be brought brought back into existence by recording whether they are enabled or not too. This particularly works well for enemies as killed enemies would appear to come back to life helping to further the illusion of time flowing backwards.

A much smaller change would be to add more visual/audio feedback to the player that they are rewinding, this is not crucial to the functionality of the reverse time mechanic and so was skipped during the development of the project in favour of time management, the time was spent on far more important aspects of the project such as other time mechanics and menu

systems. In the future for this to be implemented an audio cue to be played during the reverse time activation as well as some kind of screen effect, something resembling the old VHS rewind static screen stripes could fit the theme of the game or even showing the stopwatch design from the games main logo with the hand on the clock moving counterclockwise around the face.

5b-2. Time Travel Mechanic

Time Travel mechanic can be improved in many ways. We had a few suggestions from team players, our supervisor and people who were testing the level. The main reason why some of those ideas were not implemented was the time constraint of this project. Most of these ideas required additional time in the design phase and this particular mechanic was implemented very late.

The ability to travel between two worlds has existed in many games before. Usually, it was used as a puzzle mechanic. We had many suggestions for different types of puzzles that could have been interesting when using Time Travel ability. One of the ideas was a weapon that allows moving other things and enemies in time to solve puzzles. This weapon has been implemented, but unfortunately, I could not create an interesting puzzle that would justify using this weapon in the final build. Moreover, a puzzle might be an interesting mechanic, but it would not necessary fit into the concept of this game.

Similar to Titanfall 2 level “effect and cause” Time Travel can be used to tell a story in an interesting way. The team has never decided that story was necessary for the game. However, if we had more time available and someone would be interested in writing a story of the Time Tower, this mechanic could have been helpful.

Another suggestion was that time travel must have a bigger impact on the gameplay. In the game, this mechanic is used as a way to escape from a large number of enemies and can be used an infinite number of times. However, there are no real consequences of travelling in time. We could have added visual and gameplay changes that are a direct result of the player’s time manipulations. For example, objects destroyed in the present would not exist in the future. Although the idea was interesting, we did not have enough time to implement it.

5b-3. Procedural Room Generation

Since the rooms produced through procedural generation were very basic, there is a lot of room to improve the algorithm further.

By far the easiest advancement that could be made would be to parameterise the textures used for the floor, walls and doors. Currently these are pre-applied to the door and cube prefabs that are then stretched and positioned by the script in order to build the room. Adding public texture variables to the RoomBuilder.cs script would allow us to instead apply these to the geometry at runtime. Since there is an instance of the procedural room generator and therefore the room builder script in each scene, we then could pick and choose different textures for each level, giving them a fresh and memorable look.

Another improvement could be to add more detail into the rooms through the use of environmental structures such as pillars and tables. These would create an interesting dynamic by giving player and enemies defences to hide behind. Again, this would be a fairly simple addition to the algorithm since it would just involve instantiating a prefab in a suitable position within the room, similarly to how the enemy spawner functions. The brunt of the time spend developing this feature would be spent actually creating the structures themselves in a 3D modelling program; however free pre-made assets could also be used.

5b-4. Animations and Art Design

There was some simple graphic design work done for the games logo and weapons UI however in general none of the project team members possess any advanced design skills and so the art and animations in the game currently are free to use or very cheap to obtain Unity store assets, with very slim knowledge of implementation of these even those that exist could be greatly improved. In the future we would either need to delve far deeper into game design concepts, purchase far more advanced models and animation pack or bring a games design specialist into the team who would be able to improve the quality of the games animations and art design giving the project a much more polished look to the final product.

5b-5. Pick up Items

There are several Pick up items that have been included into the project. If future work were to be done, more pick up items would be added. With more research we could add pickups that could carry on to other levels, for example, extra lives, Armour to give the player addition health. This could give the game an extra edge on replay ability.

5b-6. Save/Load

Currently the save feature is only capable of saving the level the player is on and its current health, ideally this would more advanced and would be capable of saving all of the ammo the player has accumulated for each gun and the position of the player inside the room.

Another improvement that would be nice to have in the game would be save checkpoints once reaching/killing a boss and the ability to have floating save devices in the game that will allow the player to interact with it and save.

5b-7. Audio Design

The current state of the project has very minimal audio design and is missing audio queues in some key places. In the future we would like the game to have audio sound effects for every major action in the game, sound effects for firing bullets, bullet impacts, melee attacks, each time mechanic as well as some menu animation sound effects and button press sound effects. This improvement could greatly improve the quality of the experience of playing the game. There is music in the main menu as well as in the wardrobe but we would like there to be more music throughout the game in general. In order to do this we would like to hire some sound designers to work on the audio throughout the game.

6. References

Boccher, M (2015) *Here's how your time powers will work in Quantum Break.* [Online] gamezone.com Available at : https://www.gamezone.com/news/here-s-how-your-time-powers-will-work-in-quantum-break-3422772/ (Accessed : 11 April 2019)

Davis, G (2017) 'Procedural Level Generation in Unity for M.E.R.C. (part 1 of 2)', *Gamasutra*. Available at: http://www.gamasutra.com/blogs/GrahamDavis/20170130/290326/Procedural_Level_Generation_in_Unity_for_MERC_part_1_of_2.php (Accessed: 21 April 2019).

Hamilton, K (2018) *The Mission That Proved Titanfall 2 Was Something Special.* Available at: https://kotaku.com/let-s-talk-about-titanfall-2-s-best-mission-1788777731 (Accessed: 15 April 2019).

Mechner, J. (2008). *The Sands of Time: Crafting a Video Game Story*. [online] Electronicbookreview.com. Available at: http://electronicbookreview.com/essay/the-sands-of-time-crafting-a-video-game-story/ (Accessed 9 April, 2019).

Totilo, S. (2007). *"A Higher Standard" Game Designer Jonathan Blow Challenges Super Mario's Gold Coins, "Unethical" MMO Design And Everything Else You May Hold Dear About Video Games*. [online] MTV News. Available at: http://www.mtv.com/news/2455935/a-higher-standard-game-designer-jonathan-blow-challenges-super-marios-gold-coins-unethical-mmo-design-and-everything-else-you-may-hold-dear-about-video-games/ (Accessed 10 April 2019).

Workman, R (2016) *Quantum Break: How to Use Your Time Powers.* [Online] primagames.com Available at : https://www.primagames.com/games/quantum-break/tips/quantum-break-how-use-your-time-powers (Accessed : 11 April 2019)

7. Bibliography

Number One (2008) *Braid* [Computer Game] PC/PS3/Xbox 360. Microsoft Studios

Remedy Entertainment (2016) *Quantum Break* [Computer Game] PC/Xbox One. Microsoft Studios

Respawn Entertainment (2016) *Titanfall 2* [Computer Game] PC/PS4/Xbox One. Electronic Arts

Superhot Team (2016) *SuperHot* [Computer Game] PC/PS4 Xbox One. Superhot Team

Ubisoft Montreal (2003) *Prince of Persia: The Sands of Time* [Computer Game] PC/PS2/PS3/Xbox/GameCube/Game Boy Advance/Mobile. Ubisoft