Time Reversal
CS32420 Assignment

David Hunter

12:00 Tuesday, 1st Dec 2020

1 Introduction

The video game 'Braid' by Number None development studio is a puzzle based platformer with an interesting mechanic; the ability to undo actions by reversing time.

https://www.youtube.com/watch?v=R9Exh6uhJno

The aim of this assignment is not recreate Braid but to create a platformer with a similar time based mechanic.

As a third year assignment you will have substantial freedom to solve problems in whatever way seems best to you. **You will need to justify your solution in the report.**

1.1 Absolute Requirements

The following are absolute requirements. I reserve the right **not to grade submissions** that are not:

- Written in C#
- Written using Unity's 2D rendering and physics system. 3D models or 3D physics will not be accepted.
- Written using vanilla Unity. You cannot use any code or code plugins from downloaded from the Unity Asset store, Github or anywhere else. A few lines of code from a tutorial or elsewhere can be used with citations. As usual third party code cannot contribute to your final grade.

You may use third party art assets, with attribution. In fact, I encourage it as it will reduce the time spent on the project.

2 Requirements

2.1 Game World

The game world **must** be rendered as 2D tile based world. The game world will consist of:

- **Platforms**. These are flat static features in the world that neither players nor creatures can pass through.
- **Ladders**. These are static features in the world that the player can use to climb or descend to higher/lower platforms. The player should be able to occupy the same space as the ladder.
- **Spikes**. This are static features that kill the player on contact.
- **Goal** The location in the game world that the player is attempting to reach. Should the player stand here the player will 'win' the level.
- **Falling rocks**. These rocks take the form of a trap where rocks will fall when a player stands below them. If the rocks fall on the player the player will be killed.

In each level the game world should be larger than fits within the display. Some suitable mechanism should be created to move the camera when the player is too close to the edge of the display.

Note It isn't possible to attach components (Monobehaviours) to individual tiles on a Unity tilemap. You will need to spawn more complex tiles as separate game objects.

Hint You many need to create multiple tilemaps to implement different behaviours even for static tiles.

2.2 Main Character

The main character is the avatar of the player. The player will be able to control the main character using standard controls (e.g. WASD or arrow keys. I don't have a console style controller so please make a keyboard option available) The main character will be able to perform the following actions:

1. Walk left/right
2. Jump
3. Climb up/down (e.g. on a ladder)
4. Wind time backwards (see next section)

The player character should be fully animated and the animation should match the current actions of the player character.

3 Time manipulation

The player will have the ability to wind time backwards to correct mistakes. The amount of time that can be reversed will be limited to 20 seconds. The player should be able to see this effect played out in front of them, like playing a video backwards. All objects in the game world will be affected by this time reversal. All player characters, monsters, falling blocks etc. will be reversed also.

Time should be controlled by a single key-press, when this key is depressed normal gameplay stops and time is reversed (i.e. previous gameplay is replayed backwards) at a constant speed until either the key is released or the time has been reversed by 20 seconds. Releasing the key will cause gameplay to continue from the time point selected. There should only be one time buffer, i.e. pressing the button twice cannot take the player back more than 20 seconds.

Death should pause both the game and the time buffer, i.e. the player should be able to reverse time by 20 seconds from the moment of death.

4 Graphic User Interface

A GUI will be needed to display important information for the player. The GUI should display the current amount of time available to the player.

- Restart or Quit option if the player looses or if the player hits 'escape.' This should reset the level, quit to start screen, or resume the game depending on the selected option.
- A success message with continue (to next level) button and a quit to start screen button.
- A fail message with restart / return to main menu option.
- A start screen with instructions, level selection menu and start button.

Note: It is easy to design levels that are not winnable. You need to design winnable levels.

5 Marking and suggested implementation order

The following is a **guide only**. Final grade will depend on other factors such as quality of implementation, code, and the written report.

5.1 Minimum Implementation - Third Class

The minimum implementation does not implement the time reversal mechanic. In this implementation the player will be able to control a character that can move left/right, and jump all with simulated gravity.

- Platforms should act as solid objects that players do not fall through and the player can climb ladders.
- The player should be able to jump from platform to platform.
- The player should be able to 'win' the level by reach the end goal location.
- The player should be killed by spikes.

5.2 Almost Complete Implementation - Lower Second Class

The almost complete implementation will complete the task apart from the time reversal mechanic. In addition to the minimum implementation it will:

- The player will be able to climb ladders.
- The player character should be fully animated and the animation should match the current actions of the player character.
- The player should be killed by spikes.
- Block traps should be implemented that can fall on the players head.
- The game should have a complete GUI.

5.3 Complete Implementation - Second Class

In addition to the features of the earlier descriptions, the complete implementation will feature the time reversal mechanic as described above.

5.4 Advanced Implementation - First Class

A first class project will implement all the features as specified in this document. It will use the software design principles taught in the module to create a well engineered and flexible design. This design will be documented in the report.

6 Report

Your report should be between 4 and 8 pages long, including any screen-shots, diagrams and tables.
You can structure and format your report in any logical and easy-to-read manner (at least 11pt font). I would suggest that you include the following sections (word counts are a guide only):

Abstract. 150 words Brief summary of content and conclusions.

Introduction. 300 words. Describe the project and its aims.

Game Design. 1 page. What makes this project a game? What sort of game is it? What will motivate someone to play this game? What will such a person will get out of this game? Describe the primary mechanics of the game. Describe how those mechanics fit together to create the game. Then describe how these mechanics motivate a player to play.

Software Design. 4 pages Describe the principle components of the design. Describe how these components contribute to fulfilling the specification. Describe how the components fit together and contribute to the whole. Use UML diagrams to explain key points. What alternative designs do you consider (or try) and what are the pros and cons of these different choices?

Testing. 1 page. Test your software using appropriate tests. For example Unit tests or user testing.

Discussion and reflection. 1 page. This section should answer the following questions: What are the primary strengths of your project? What are its weakness? What have you learned during this project? What would you do differently next time? If during self reflection you have identified an issue, e.g. time management, what actual steps could you take to address this issue?

7 Submission

Both the report and the completed game should be submitted via Blackboard.

Your project folder containing your Unity project files, C# code, Unity Scenes, Graphics etc. should be packaged in a **ZIP** file and submitted to Blackboard.

Your report should be saved as a PDF and submitted to Turnitin.

Your game should be built and packaged as a WebGL webpage and made available on your public_html pages in your personal directory. Don't forget to follow the instructions from IS on how to make files public. (http://users.aber.ac.uk/en/) **You must provide a link in your report!**

7.1 Submission Check-list

Required files	Format	Submission point
Unity Project files	Zip	Blackboard assignment file submission point.
Report	PDF	Turnitin
Packaged Game	Unity WebGL	public_html/CS32420/

12:00 Tuesday, 1st Dec 2020 As per departmental and university policies, late submissions will be awarded zero. Early submission is advised, multiple submissions are permitted. If you encounter technical issues please submit what you can, as soon as you can. Email David Hunter dah56@aber.ac.uk as soon as possible to explain your situation. David Hunter cannot decide if a late submission will be accepted but can pass on your explanation to the exam-board for their consideration.

8 Assessment Criteria and Marking Specification

This assignment is worth 50% of the marks for the module CS32420.
The main criteria of the marking specification are divided into two main sections:

1. your implementation of the game. 50%
2. your report. 50%

All submitted work will be graded according to the criteria in the Department of Computer Science student handbook.

8.1 Special Circumstances

If there are any circumstances that affect your work, please visit http://impacs-inter.dcs.aber.ac.uk/en/cs-undergraduate/resources/special-circumstances to see what to do. For some situations, a coursework extension is sufficient. In either case, it is the year tutor (Neil Taylor, nst@aber.ac.uk for third year Computer Science students) who will decide the appropriate action to take in your case.