Game Development Report, Adrian Tukendorf [adt24]
Assignment; CS32420 Time Reversal, Platformer Game

https://users.aber.ac.uk/adt24/

1. Abstract

I created a requested 2D platformer game, with all specified in the assignment brief features, including time reversal. My game title is AstroBoy. In this document, you can find all information about my game, its storyline, game mechanics, details from the side of software engineering, testing, and my own reflections and experience during work on this project.

2. Introduction

The project aim was to create a platformer game using Unity's 2D rendering and physic system accompanied by several C# scripts. The goal of the game is to complete the level without dying. Although the player can reverse time up to 20 seconds, even in the event of death. Implementation of the time-reversal concept was the main challenge of the project. Every player's action and its environment had to be reversed during the press of the button. It should create an illusion of time going back when the key is depressed. Everything should be happening on the screen and the player can admire all game objects going back in time. The gameplay is going back to normal when a specific key is released. In this project I had to create a **2D tiles** based map consist of: **Platforms** – flat static features non-penetrable by any sprites, **Ladders** – static features that the player can use to climb or descent to higher and lower levels of the map, **Spikes** – static feature able to kill the player on contact, **Goal** – the location in the game world that the player is attempting to reach if done so the player is winning and able to proceed to next level, **Falling Rocks** – form of a trap falling from the highs and on contact kills the player. In my opinion, the assignment aims to teach students how to create a 2D Unity based game from scratch. Using only vanilla Unity without any extra plugins or parts of code accessible from the unity assets store. This creates a great opportunity to face lots of game development problems and overcome them. During the process of development, the student can learn the basics of C# language, usage of Unity Editor, and good software engineer design practice.

3. Game Design

A. Story - This section describes the details for a 2D platform game where AstroBoy, the main character, tries to reach a goal-point at the end of the world full of dangerous traps.

B. Character - AstroBoy is a time and space traveller who explores different words and for his luck has an ability to reverse time and correct mistakes. AstroBoy is a born traveller, he never had a real home. His main motivation is to find a planet where he can stay for longer. Although most of the planets he visits are very dangerous, that's why he keeps tirelessly looking and exploring deep space to find a peaceful home. With every next level, he has to face-up waiting for him traps and spikes. He has an amazing special power that keeps him alive– Time Rewind, AstroBoy can rewind time and correct mistakes.

C. The theme of the game is quite nostalgic because AstroBoy travels alone, there is no other living soul on the planets he visits. But every step is moving the main character closer to his dream home.

D. Gameplay

D.1 Goals

The overall goal of the game is to help lonely AstroBoy find his dream home in an empty and dangerous deep space. In more detail, the player has to figure out how to move through the map, avoid spikes, falling stones and have to be careful to don't fall from the platforms into an endless void. Finish the level by reaching the portal to the next planet.

D.2 User Skills

- Strategic approach to find the correct way through the map
- Good reflex during jumping from platform to platform, dodging falling rocks
- Ability to use the keyboard for control of the main character

D.3 Game Mechanics

- The player can move around the map by jumping from one platform to another platform if the player will fall from the platform that results in a failure.
- The player can move up/down using ladders
- The player can jump by pressing a space bar.
- The player can reverse time up to 20 seconds backwards to correct mistakes.
- Time can be reversed after death; counter stops when the player dies.
- The player can use 5 health points before dying, they are displayed in the form of pixel hearts on the top left corner of the screen. Zero hearts results in death
- If the player is falling longer than 5 seconds it results in the death of the player.
- If the player touches spikes, the player receives 1 point of damage.
- If the player touches a falling rock, the player receives 1 point of damage.
- Levels remain locked if the previous one has not been completed.
- The next level is unlocked when the player successfully attend the Goal
- The goal is to complete the level by entering the portal door on the end of the map.
- The player can give up and commit suicide by pressing the Escape key.
- The player can reach the destination without a way out, the only way to proceed with the gameplay is committing suicide or reversing time.

Conclusion

Game mechanics and a set of rules together create a game where the player can move freely around the map, encounter challenges, and finally achieve a goal. The motivational factor to play the game is by achieving the goal and unlocking the next levels but also the possibility to fix mistakes committed in the past. Giving an example where players are almost at the end of the map, seconds away from completing the goal, and fails by accident. Time reversal feature saves frustration to the player, if the feature is used correctly the player can proceed with the gameplay and level up successfully. It's very helpful especially for beginning gamers who without this feature would encounter disappointment of failing the level and in effect lose motivation to play more. The player can also learn from mistakes and try different approaches to the encountered problem. As a result, the player gets better and is not losing more health points. Players can compete with each other, the one who completed the level and lost fewer health points wins. Elements like spikes, rocks, or ladders are enriching the gameplay by showing the interaction between the player and the game world.

4. Software Design

This section describes the technical side of my game. I will describe each of the classes, their dependencies and functionality. (UML diagram uploaded separately, it was too big to fit here)

Class PlayerMovement

This class is responsible for player movement including jumping, ability to climb ladders, death by falling, suicide feature when players give up, overall player's health management, and detection of rocks colliders. This class sends information to the animator inside Unity Editor to make possible playing the right animation depending on the player's state. It is also responsible for loading the next level in the case when the player reaches the goal.

Void Start() - is called only once on the frame when the script is enabled, in my case it is used to initialize the RigidBody2D component of the player and TimeManager object to allow me to use it later on in the code.

Void FixedUpdate() – this function is called every fixed frame-rate frame, I decided to use this function because code inside adds force to Rigidbody2D, it is recommended because Update() function is called more often and it might cause malfunction of the application by adding force to the component twice or more during one frame.

It is responsible for the player's movement by taking input from the keyboard and adding velocity to the player's sprite accordingly. If the player collides with the ladder and arrow-up is pressed it adds force to the player's Rigibody and disables gravitation to fulfil the game requirement - the player is able to climb ladders. It also checks if time is not reversing or current health is more than zero. If time is reversing, the movement of the player is disabled. It sends key information to the Animator component, isClimbing – Boolean – true if the player is climbing, Speed – Float – informs the Animator what direction is the player heading, isGrounded – Boolean – true if the player touches the ground. There is also a check, to make sure the player is facing the direction it moves, player is flipped using Flip() function. I used code from the practical to flip the player, I couldn't find any better way to do it.

Void Update() – this function calls functions: *Jump()* – responsible for the ability to jump by adding force to players RigidBody if the player is grounded and a suitable key is pressed. *DeathByFalling()* – it checks if the player is not touching the ground for more than 5 seconds, it kills the player if the 5-second limit is exceeded to avoid infinite falling. *Dying()* – this function checks if the player's current health value is lower or equal zero, if yes fail screen is displayed. *Suicide()* – after pressing the escape key, the player's current health is decreased to value zero.

Void Damage(int) – decrements the passed value from the player's current health. It is called when a rock hits the player or the player touches spikes. Here I wasn't sure if the player should instantly die after receiving damage or being killed as the result of encountering the trap couple of times. To make the game more enjoyable I've decided to implement an additional feature – a health bar. I hope it fulfills game requirements from the brief, damage amount can be changed from 1 to 5 to kill the player instantly on contact.

void OnCollisionEnter2D(Collision2D collision) – using tags specified in the Unity editor, players collider can detect several objects, Rock/2/3 or FinishPoint. If the rock is detected function *Damage(1)* is called. If the finish point is detected win menu is displayed and game progress saved using PlayerPrefs. I could have used Serializable to save player progress but for a simple game, PlayerPrefs seemed just fine. Although, it is not an ideal solution because the main purpose of this function is to save the player's preferences and it is easy to edit in the game files.

Class CameraAndGUI

responsible for smoothly changing camera location to follow the player, smoothening can be edited in Unity Editor. It also displays the current health of the player by displaying different sprites accordingly to the player's current health. Due to AstroBoy's Gameplay, it wasn't suitable to implement "camera push" – when the player is approaching the edge of the screen camera moves. I did try this solution but the one implemented was making gameplay much more enjoyable. In my opinion, it is better to slowly reveal the map when the player is approaching the unknown part, especially when the player is climbing a ladder or jumps down the platform to lower levels. *Mathf.SmoothDamp()* is used to smoothen the camera movement. It changes the value gradually over time to the desired goal, instead of instant movement which feels artificial during the Gameplay.

Class DisableAcceleration

This mischievously looking class is used to stop objects' acceleration after reaching a specified maximum. Rigidbody properties are preserved but limited. I created this script to solve my problem with the implementation of the falling rocks. The script is assigned to every rock object, the velocity of the rock is limited, and they are not becoming projectiles.

Class Rocks

It controls falling rocks' behaviour. It was a challenge for me to get this trap working with the time-reversal feature and I had problems which I had to overcome. In my interpretation rocks are GameObjects with Rigidbody component, they are constantly falling, and the player has to dodge them creating a trap. When a player stands below a rock, he receives damage which in result can kill the player. I was testing many implementations but the one working for me is that the rock is falling from the upper part of the map and when it touches the ground (in my case created for this reason object collider) it is being teleported back to the starting position *Spawn/Spawn2/Spawn3.* The problem was that rocks were constantly accelerating, changing a rock into a projectile. I overcome this problem by limiting the velocity of the rocks using the

DisableAcceleration script. This class is a component of the object, with the collider which represents the end location of the rock, that should be teleported back to the start position. Every rock is distinguished from each other using Tags. OnTriggerEnter2D function detects objects and depending on their tags, teleports the object to the corresponding spawn point.

Class Spikes

Spikes are implemented by using a separate tilemap with a component *Spikes*. Tilemap is a collider if it detects object player then function *player.Damage(1)* is being called.

Class Grounded

This class helps the *PlayerMovement* class to determine if the player is grounded or not. If the collider detects an object with the tag *Ground,* it is changing the Boolean value of the variable *isGrounded* to *true* value in the *PlayerMovement* class. This component is assigned to the box collider on the AstroBoy sprite.

Class TimeManager

This class is responsible for the feature of time rewinding. In my project, all positions, states, and actions of objects in the game are being recorded with every frame update. Data are stored in the ArrayList datasets. There are 4 ArrayLists recording player data: playerPosition – record transform component of the player, *climbingInfo* – records Boolean indicating that player is climbing, *direction* – records *inputHorizontal* variable which indicates what side is the player currently facing, *playerHealth* – records integer of players current health. There are another 3 ArrayLists, recording data about the position of each rock. Firstly, all variables and datasets are initialized in the *Start()* function. *FixedUpdate()* function checks if the time is reversing, if it is not reversing then it checks if the current health of the player is higher than zero. If all of these conditions are met, code works, and with every frame records the position of all elements described above. If time is reversing (*isReversing == true*), the current object's components and variables are being overwritten by the values stored in the ArrayLists. Every next frame update iterates through the ArrayLists and values that have been read are being removed at the same time. Check for the "R" button being pressed is made in *Update()* function, together with the limitation to rewind time no longer than 20 seconds. Also, it checks if the player is still alive, if not timer stops, and the recording of data is stopped. Thanks to this, it is possible to rewind time even when it is been longer than 20 seconds after the player's death. The textBox is being updated to show on the screen how many seconds the player has left to rewind.

Class UIManager

This class is mainly responsible for loading and reloading scenes. Functions from this class are called in PlayerMovement class to progress to the next level, MainMenuUIManager class to load selected levels, start a new game, or quit the game. Class tracks what scene is now active and assigns the index number to the activeScene variable. It is used to load the next level by simply increment this variable (activeScene++), then uses method SceneManager.LoadScene(activeScene). Function NewGame() not only loads the first level but also resets the game progress by assigning the index number of the first level to the PlayerPrefs-" level".

Class MainMenuUIManager

Responsible for hiding and displaying elements of the UI in the Main Menu in accordance with user intentions. If the player chooses option Levels in the Main Menu, ChooseLvl() function is called, firstly it checks if the player has any progress saved (PlayerPrefs.HasKey("level")). If yes, then UI elements of the main menu are being deactivated and the level menu is displayed. Buttons to choose

the levels appear only if the player unlocked the level. The switch statement checks what level is the highest unlocked and accordingly, displays the buttons loading levels.

5. User Testing

Feature to be tested	Pass test criteria	Result of the test	Does it pass?
Does the game start in Safari browser?	Game starts in time shorter than 10 seconds.	Game starts in 5 seconds; main menu is displayed.	PASS
Is the level menu not available during first start?	Level menu is not displayed during first run of the game.	Button is pressed; menu doesn't show	PASS
Is the level menu displayed after unlocking Level 2?	After choosing Levels button in the main menu, level menu should be displayed with all unlocked levels and options to go back to main menu.	After finishing first level, second level is unlocked. Level menu is displayed with two levels to choose and Go back button.	PASS
Is the first level loaded after pressing New Game button?	Level should be loaded.	Level is loaded.	PASS
Does the GUI display correctly?	Health indicator and timer should be displayed in top left corner of the screen.	Health indicator and timer are displayed.	PASS
Is the sprite moving accordingly to pressed buttons?	Left arrow – sprite moves left. Right arrow – sprites moves right.	AstroBoy is moving accordingly to the pressed keys.	PASS
Does the sprite jump after pressing the Space Bar?	After pressing the Space Bar, sprite is instantly jumping.	Space bar causes AstroBoy to jump.	PASS
Does the sprite climb the ladder?	AstroBoy should climb the ladder when when key-arrow-up pressed and player is on the ladder.	AstroBoy climbs the ladder.	PASS
Does the Time Reversal feature work correctly?	When R-key pressed time should go back, displaying all the player animations accordingly. Objects in the game: falling rocks, health indicator should be impacted.	During holding the R-key - Time is rewinding, it impacts the player, including animations. Rocks and health indicator are also impacted	PASS
Does the time reversal stop after reaching maximum of 20 seconds?	Time reversal should stop, and normal gameplay continued. Timer should reach zero on the end of the reversal and start incrementing afterwards.	Time reversal stops after reaching 20 seconds, displayed timer works correctly.	PASS

Do the spikes give harm to the sprite? Is the health indicator corresponding accordingly?	After contact with the spikes, one health heart is removed.	Spikes give damage to the player; health indicator works as described.	PASS
Is the camera following the player?	Camera should follow the player with smooth movement.	Camera follows the player; movement of the camera is smooth.	PASS
Does the falling rocks fall and give damage to the player?	When rock fall on the player, player loses one health point.	When rock hits the player, one health point is being removed; rocks fall.	PASS
Is the fail menu displayed after players death?	Player should die when all health points are being used. Fail menu with Try Again/Main Menu displayed.	After using every health point, “You died.” Information is displayed with two earlier described options.	PASS
Is the win menu displayed after reaching a goal?	After reaching the goal win menu should be displayed with two options: Next Level or Quit	After reaching the goal point, message “Congratulations!” is displayed, two options earlier described are displayed.	PASS
Does all animations display correctly?	During walking/jumping/climbing sprite should play suitable animations instantly. Player should be flipped during change of direction.	All animations are being displayed correctly; player is facing right direction during movement.	PASS
Is the game closing after pressing Quit button in the main menu?	Game should stop working after pressing Quit button.	Game stops working, grey background is displayed in the browser.	PASS

Disclaimer

The game was tested on MacBook Pro 2017, processor: 2.3 GHz Dual-Core Intel Core i5, memory: 8 GB 2133 MHz LPDDR3, graphics: Intel Iris Plus Graphics 640 1536 MB, operating system: macOS Big Sur 11.1 Beta (20C5048k)

Using Safari web browser Version 14.0.1 (16610.3.6.3)

6. Discussion and reflection

What are the primary strengths of your project?

The biggest strength for me is that I managed to implement all required features especially that it was my first game, and first-time using Unity and C#. I managed to add some additional features like a health indicator or that the player is killed when falling from the platform.

What are its weaknesses?

In my opinion, the biggest issue is efficiency. I used a lot of Update and FixedUpdate functions, although the frame rate is very good, I can hear my laptop's fan is speeding up. It might be also caused by TimeManager which almost constantly with every frame record position of every object which moves. I could have work more on my design, for example, PlayerMovement class is responsible for much more than only the player's movements. When the game was growing because of a lack of my Unity knowledge at the beginning of the project, it was just easier to implement described features this way.

What have you learned during this project?

I learned a lot; I think it was a great opportunity to learn how to build a 2D game from scratch in Unity. I learned some basics C# scripting which I think is very interesting, especially with Unity because you can see the effects of your work almost instantly, it is very motivating. I am really happy that I chose this module. The knowledge I gained during this project is will be very practical in future projects.

What would you do differently next time?

I would improve my software engineering design – definitely. The efficiency of the game, to use fewer hardware resources. I would add more levels, make them bigger, and storyline of the game more visible in the Gameplay. I would add monsters and very possibly multiplayer to put more emphasis on player competition. Collectible points, like coins or diamonds to motivate the player to reach higher scores by collecting them.

Self-reflection

I think it was a great project, I really enjoyed the process of game development. I learned a lot of useful skills. Therefore, I identified an issue with planning at the start of the project. I didn't really know where to start. I believe that if I would have more experience, my game design would be definitely improved. I think my time management was good, I spent lots of hours reading, watching lectures, and additional tutorials to prepare myself for the assignment. I was also working on the assignment without a hurry, which helped me to explore, understand, and learn about Unity tools. I wish I had more experience with Unity at the start of the project, that would help me create a much better-quality game. Also, I am not sure if I got the Falling Rocks interpretation right. My rocks give damage to the player, instead of killing instantly. It's worrying me that I got it wrong, even that the assignment brief stands that it's a third-year project and we will have substantial freedom to solve problems.

Credits:

Creating the game: Adrian Tukendorf

Free Assets: Aiden Art; Free Pixel Space Platform Pack

Source of the assets: https://assetstore.unity.com/packages/2d/characters/free-pixel-space-platform-pack-146318