Burning Choices

Ethan Seiber, Jamel Warren, and Hunter Haislip

Abstract

This project is a 2D game named "Burning Choices". The game will primarily focus on the adventure genre. It will include multiple levels and branching story. The game took inspiration from Bandersnatch, a fictitious game in the Netflix original series Black Mirror that also included branching story line. The target audience will be those that enjoy classic action arcade style games.

1. Introduction

Burning Choices is a 2D adventure game that operates around theme of angels, demons and humans. The end goal of the project is to tell a story that branches depending on the actions of the player. The story will follow the player as they find their way through our world overcoming any obstacles in their way. In doing so they are given the ability to control their own story. The audience for this project would be those who enjoy retro style games. The main goal is for our audience to play the game and keep coming back for more thanks to our game leaving them feeling satisfied and entertained. We chose this project thanks to our different interests ranging from a love of video games to things we enjoy in our own lives and would like to create ourselves.

1.1. Background

Throughout the paper these terms will be used in describing certain aspects or situations in the game. 2D graphics, branching storyline and linear storyline. 2D graphics refers to the dimensions of of the art style within the game i.e. 2D graphics is short for 2 dimensional graphics. Mario for the nintendo 64 is a great example of 2D graphics. Branching storyline refers to the story structure and how the story of the game is built. The story that is produced depends on the choice of the player. This means the player could end up with a completely different ending or story leading to the ending depending on what they do in the game. Linear storyline also refers to the story structure within the game. A linear storyline is much like reading a book. From beginning to end the story is static and unchanging with only one end. Ethan and Jamel chose this project to hone their game development skills, while Hunter was more interested by the story aspect.

1.2. Challenges

Some of the challenges we suspect to run into include programmatically creating obstacles like walls and items the player can interact with, giving the player the ability to move around each level and the creation of many levels that are connected so the player may move between levels. We know how to gather a user's key inputs, but we need to make the keys perform actions like move the character. This will be the first time we have worked with moving an object on screen using our own code. Next we have to overcome how objects like walls and items act within the program because a wall is different than an item. A wall the player can't pass through, but an item they could. Saved games are a necessity because it records the player's progress within the story thus far and stores it away so that they may access it later to load their game. The last expected challenge is to create our different story lines and make sure they lead to an ending. To address these challenges we will need to do some research on useful libraries to help provide pre- defined functionality or create our own functionality. Things we have found to help us are the C# classes Canvas, Image, BitmapImage and Grid. The Canvas class has allowed us to create the levels for which the player will travers. The Image and BitmapImage classes help us give graphical pictures to our walls, items and level. The Grid class allows us to be able to create an inventory system that will help the player keep track of items collected thus far within the game.

2. Scope

The project will be finished when we have at least three in game levels, a linear storyline and can save. The stretch goals we have set for this project are the branching storylines and additional game levels.

2.1. Requirements

The functional requirements for this project were gathered from our experiences in other games.

Use Case ID	Use Case Name	Primary Actor	Complexity	Priority
1	Starting the Story	Player	Easy	1
2	Game Saving	Player	Med	2
3	Exiting Game	Player	Easy	2
4	Load Game	Player	Med	2
5	Character Movement	Player	Easy	1
6	Items	Player	Hard	1

TABLE 1. USE CASE TABLE

2.1.1. Functional.

- On load up of the game. Start on the title screen.
- The player will be able to save the game.
- On the title menu the user can choose to load a game, start a new one or quit.

2.1.2. Non-Functional.

- Recoverability- If the game crashes then it can be rebooted and the user can restart their game using a loaded save.
- Maintainability- Code is stable so that we can easily add new features with few to no problems.

2.2. Use Cases

Use Case Number: 1
Use Case Name: Starting the story.
Description: The user is on the title screen and wants to start the story of the game.

1) The user will click on New Game to start a new story.
2) The game starts at the beginning of the story.

Use Case Number: 2
Use Case Name: Game saving
Description: The user reaches a point in the game where they wish to save their progress.

1) The user will hit the Esc key to pause the game.
2) The game environment will pause.
3) A list of menu items will pop up.
4) The user will select Save Game in the pause menu.
5) The game will store the player's story data.
6) The game will un- pause and continue where the user left off.

Use Case Number: 3
Use Case Name: Exiting Game
Description: After playing for a while the user will want to quit the game.

1) The user will hit the Esc key to pause the game.
2) A list of items will pop up.
3) If the user selects Quit the game will exit and stop running.

Alternative: User is on the title screen

1) The user will navigate to the bottom of the title screen.
2) The user will click quit and the game will exit.

Use Case Number: 4
Use Case Name: Load Game
Description: The user is on the title screen and wishes to continue where they left off in the story.

1) The user will click the menu item "Load Game".
2) The game will open the file holding the saved data of where the user last saved.
3) The game will load to the point where the user last saved in the story.

Use Case Number: 5
Use Case Name: Character Movement
Description: The user is in a game level and wishes to move within the level.

1) The user presses one of the keys w,a,s,d on their keyboard.
2) The character the player controls will move in the direction of the button pressed.

Use Case: 6

Use Case Name: Items

Description: To complete quests the player needs to collect items.

1) Player has to collect items.
2) The player walks onto an item.
3) The item is collected by the player.

2.3. Interface Mockups

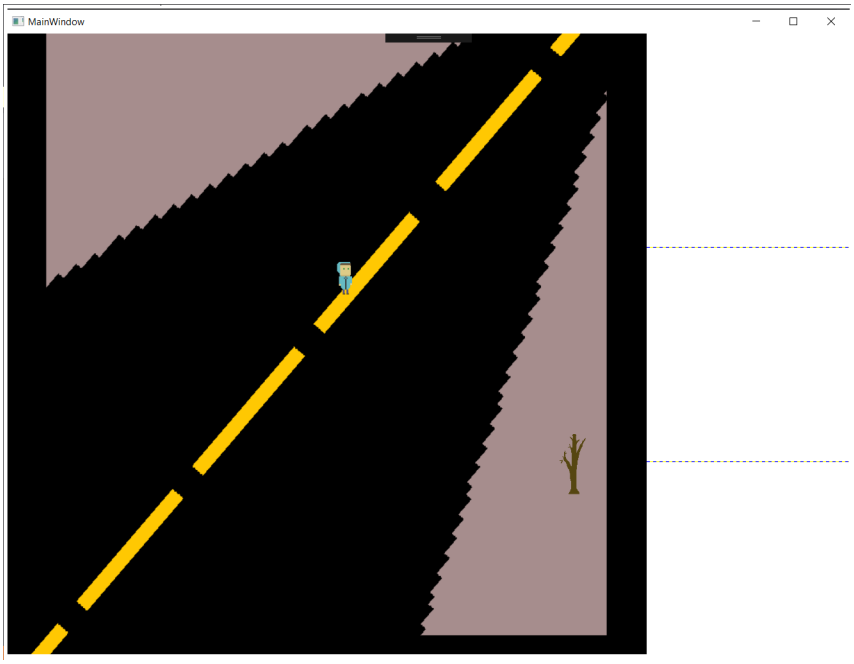

Figure 1. The gameplay so far

This is a very early version of our level design. This particular scene will involve a lady the player can talk to and she will give the player a quest to collect items. To the right of the game is the inventory system for item collection. The black bars are walls that the player can not go through. At the top of the screen is how the player will transfer to a new level.

Figure 2. A layout of the Title Screen

This is what we want our title screen to look like once we are finished with the project except we plan to replace start with new game. This image is the first thing that will pop up when you run the game. It shows you what we envision use cases 1 and 4 to be like. The Resume option, case 4, will allow the player to load a game to continue the story where they left off and the Start option, case 1, will begin a new game from the beginning of the story. The Quit option pertains to use case 3's alternative in which the user is on the title screen and wants to quit the game.

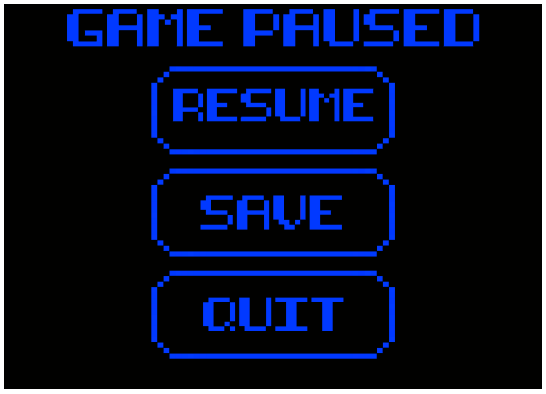

Figure 3. A layout of the pause screen

This is the current version of our pause screen. The screen will give you the three options to choose from. The Resume button will un-pause the game and continue the user where they left off. The save option is use case 2 which will record the user's current progress within the story and save it to a file to be loaded later. Then at the bottom of the list is quit which is use case 3. This will close the game entirely.

3. Project Timeline

The timeline for this project starts with the requirements phase. It will last from January 2019 to January 2019. This phase is simple and took little time to complete. We took looked at things that were convenient and useful in other that can be applied to this project, so that the user can have the same convenience with this game. An example of this would be a saving feature.

We are currently in the design phase. This phase runs from February through March.

1) We deliberate on the type of gameplay we will have.
2) Discuss the story.
3) Decide a theme for the whole game.
4) Deliberate on some design patterns to use.

Next is the implementation phase. We plan for this to last from February through April. In this time we plan to have these things finished.

1) Levels designed (April 2019- April 2019)
2) Obstacle class made (March 1, 2019- March 23, 2019)
3) Inventory system made and implemented (March 25- April 14, 2019)
4) The story implemented (April 15 2019- April 21, 2019)

Next is the verification phase and it will last from April 21, 2019 through April 22, 2019.

Lastly is the Maintenance phase which will last from April 22, 2019 to April 24, 2019.

It's still hard to tell whether we budgeted our time well. We have the majority of the gameplay implemented and waiting to be used, but there are still many things like the main menu, pause menu and the story/ levels that need to be finished. The main menu and pause menu have their own functionality that is separate from the gameplay that has yet to be defined. With these tasks taken in consideration with the fact that the deadline is close then we probably did not budget well.

4. Project Structure

Currently we have decided to use an Abstract Factory design pattern to define and create any objects we will use within the game such as walls and doors.

The second pattern we are using is the Observer pattern. This keeps the inventory system up to date.

4.1. UML Outline

4.2. Design Patterns Used

Currently we are using the Abstract Factory design patter as one of our patterns. It allow us to create obstacles like walls, doors, items and characters. The second pattern we used was the observer pattern. We needed a way to update our inventory grid when the player collects an item. Using a subject we collect the item and send it to the observer which puts it in the inventory grid.

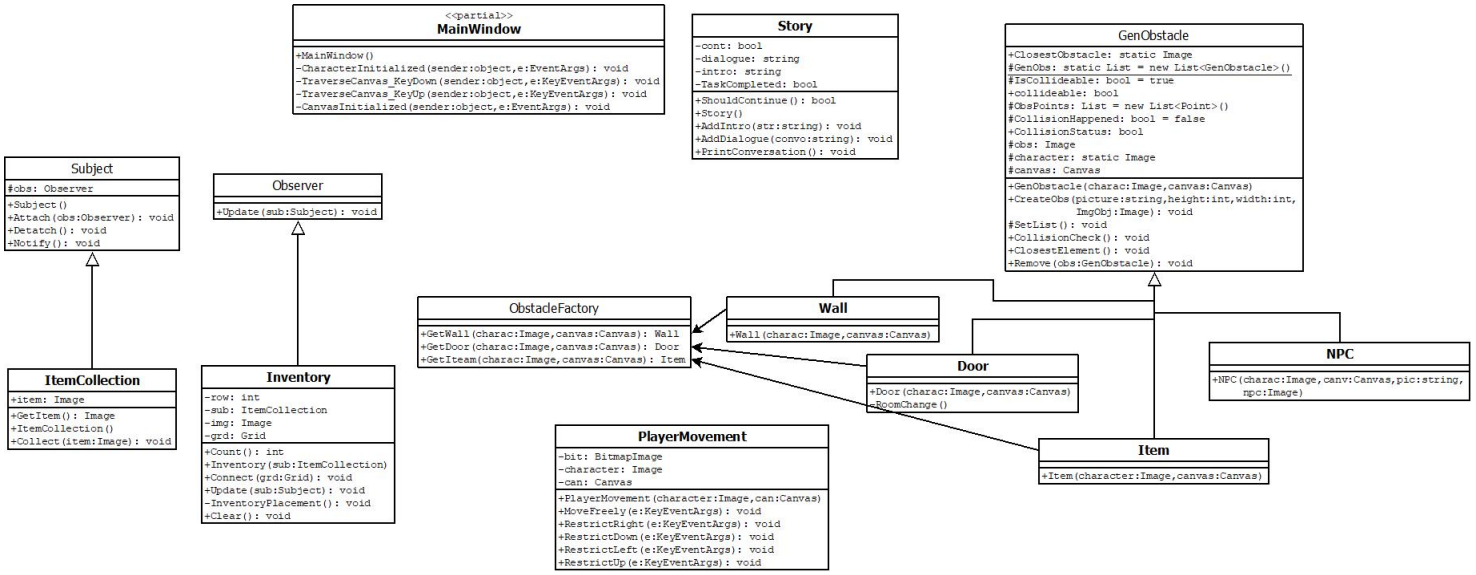

Figure 4. This is the current UML are using for our project. It is not finished for there may be a need to create another class for transferring into new levels. The Story class is how we plan to tell the story of the game. It is currently being developed. The PlayerMovement class is how the player moves throughout the game. The MoveFreely function allows the player to move around un- impeded. The functions with Restrict at the begining of the name restricts the player's movement in a direction that is defined at the end of the function name. The GenObstacle class is an abstract class that defines common functionality for all obstacles that will be used within the game. Individual obstacles such as: walls, doors, items and other characters inherit from GenObstacle to share in the functionality. Each obstacle has its own way the player will interact with it. For example the player can't walk through a wall. The Subject class defines common functionality for any derived class to inherit. The ItemCollection class inherits from the Subject class and works with the Inventory class. The Inventory and ItemCollection classes work together to form an observer and subject relationship. The ItemCollection class will collect an item obstacle when the player walks into an item and then send it to the Inventory class to be put into the player's inventory.

5. Results

The project's gameplay functionality is nearly complete. This means the player can collect items to the inventory and the player can't travel to places they aren't allowed to. We have the capability to create the story within the game as well as little tasks to complete in the game, but we have not taken the time to really add them.

5.1. Future Work

There are many goals we haven't made thus far. We haven't implemented the linear story and levels. Then there are the main and pause menus. These were the things that we decided would take the least amount of time and thus were slated to be finished last. Things that we would add if we had more time and no other tasks to get through would be gameplay. Perhaps we would do a hedge maze within the game. Then we might make the story less of a pattern. Currently we have the story operate in a pattern of 3. The player does interacts with something, they get a quest and they complete the quest and leave for the next level.