Salty Souls

Trever Hall and Wade Wakefield

Abstract

The project is a 2D platform game programmed in c-sharp in a WPF application. It is a throwback to the art and design style of games like the original Castylvania. It will have basic *W,A,S,D, spacebar* controls for movement and *J* for attacking.

1. Introduction

This project is a video game. It is a two dimensional platformer. It will include the ability to walk, jump, and attack enemies. The background is set, and the enemies spawn from the edges of the screen. The user is to fight those enemies for special drop items or in-game currency to buy upgrades that make it easier to get further in the game. When the player dies, they arrive back at the market to buy special items, and upgrades. Currency received is added up across every time played.

The motivation for this is that we grew up playing video games, and we have both always wanted to create a game like the ones we played as kids. The game will be pixel are It is not going to be super technical for today's standard of video games, but it is definitely going to be a challenge since neither of us have attempted something on this scale and with such complexity.

1.1. Challenges

The main challenge we believe is creating interaction between the character and the world around, and animating everything. Precise timing is needed for both to work correctly. For example say you are attacking at the split second the enemy moves, the enemy will not exist on the screen because it is being redraw to show movement, so the attack will not hit it. Solving issues like that and timed animations in general are what we believe to be our biggest challenges.

2. Scope

At the bare minimum to consider the project complete, we want to have one level (determined by the tile-set and background used) that the user is able to fight 3 different types of enemies on. Characters will have an idle and at least one attack animation. There will be a market where you can buy upgrades(2 types) and items(2 types) to use in the next play of the game (accessible from the market by button). Then finally a *pause menu*, which is going to work as the main menu when the game starts up, to quit or assess important information and stats.

2.1. Stretch Goals

2.1.1. Multiplayer. Support for local multiplayer for two people.

2.1.2. Level Editor. Allows the user to use tile sets to create own level and background.

2.2. Requirements

We sat down together and discussed the requirement that we thought were necessary and they are listed below and elaborated on as needed.

2.2.1. Functional.

User will be able to use the main menu.
User will be able to start a game.
User will be able to play and fight enemies.
User will be able to visit the market.
User will be able to earn in-game currency.
User will be able to spend in-game currency.

2.2.2. Non-Functional.

Capacity - one player, unless multiplayer implemented (2 players)
Reliability - money will not disappear when you die
Recoverability - when shut down, should be able to start up with a previous stuff earned

2.3. Use Cases

Use Case Number:	1
Use Case Name:	Use Main Menu
Description:	Is where the user chooses the action they want to do. They can either play a run of the game, see stats, quit, or see controls.
1)	The user navigates to the menu, or is greeted by it at programs start.
2)	User clicks desired action
3)	The user is redirected to what the button described
Use Case Number:	2
Use Case Name:	Start a game
Description:	The user should be able to start a run of the game.
1)	The user navigates to the market.
2)	The user hits the *go* button to start a run of the game.
Use Case Number:	3
Use Case Name:	Visit the market
Description:	The user should be able to go to the market.
1)	The user navigates to the menu screen
2)	The user clicks on the market button to go to the market screen.

2.4. Interface Mockups

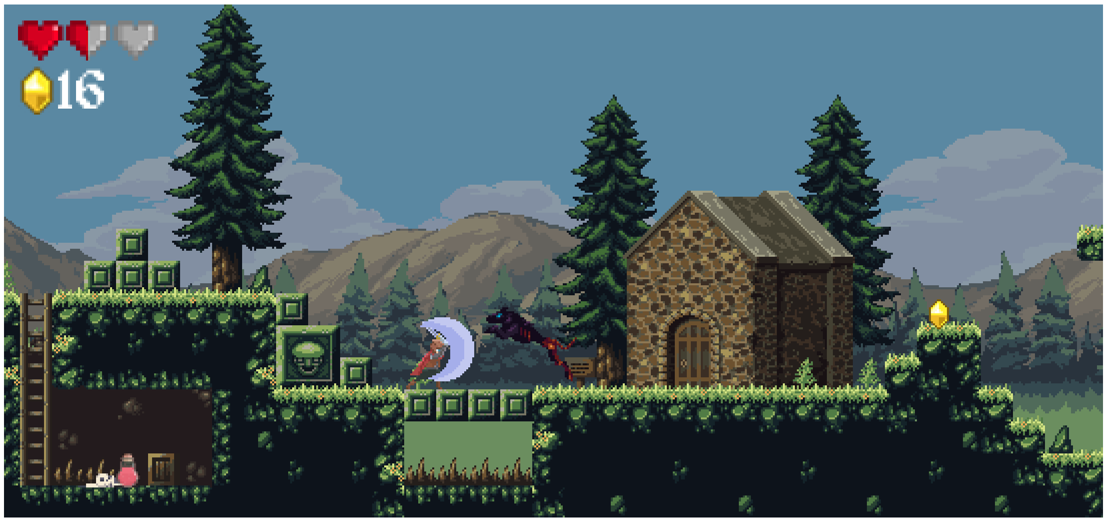

Figure 1. This is an example of the UI of a run of the main gameplay.

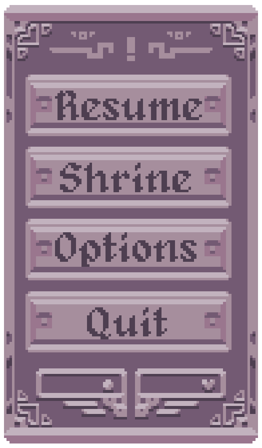

Figure 2. The pause menu for when in gameplay.

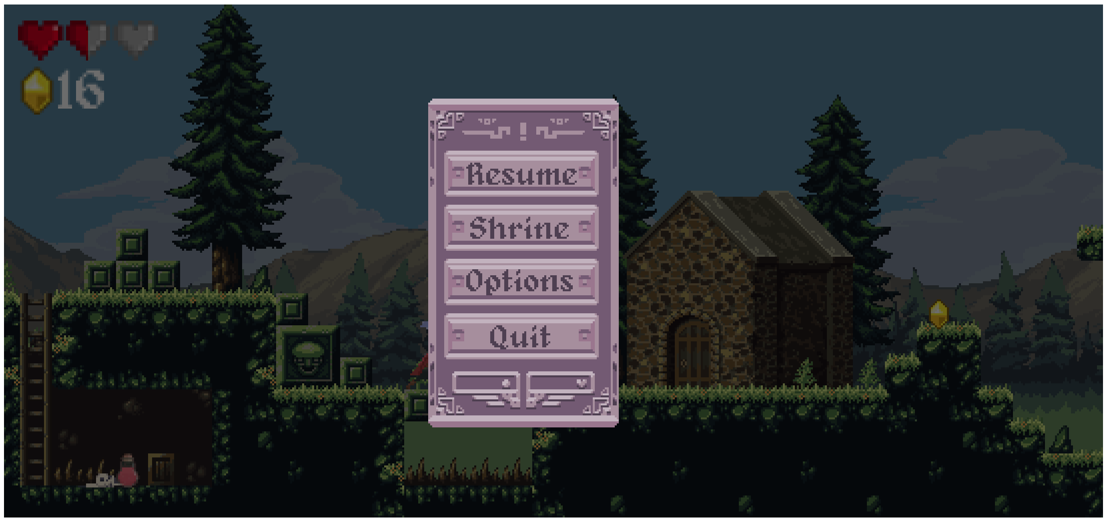

Figure 3. The pause menu over the gameplay.

3. Project Timeline

September 4th: Time based physics engine created and attached to Entities within GameWorld.
November 4th: Was able to create a moving image in the bitmap.
November 14th - 15th: Add all image resources needed, change image based on left/right direction, and have animated movement.
November 16th - 18th: Able to pause the game and character can jump.
November 19th - 20th: Stages created with collision bounds.
November 21st: Have a moving enemy(s).
November 23 -25: Health system figured out and able to damage enemies and they can damage the user.
November 26 - 28: Stat changes and the market added.
November 30 - 2: Add stage changes and troubleshoot/ add finishing touches.
Until December 6th: Get everything working and add anything overlooked.

4. Project Structure

At first, this will be a little empty (it will need to be filled in by the time you turn in your final report). This is your chance to discuss all of your design decisions (consider this the README's big brother).

4.1. UML Outline

Show the full structure of your program. Make sure to keep on updating this section as your project evolves (you often start out with one plan, but end up modifying things as you move along). As a note, while Dia fails miserably at generating pdfs (probably my fault), I have had much success with png files. Make sure to wrap your images in a `figure` environment, and to reference with the `ref` command. For example, see Figure **??**.

4.2. Design Patterns Used

Make sure to actually use at least 2 design patterns from this class. This is not normally part of such documentation, but largely just specific to this class – I want to see you use the patterns!