Jonathan Lewis | Ian Davies | Jacob Schwartz

Design Patterns Final

Team Eagle Plaid

Game Description

Our game is based on Adventure Time, developed in MonoGame, XNA and C#. The game supports 1 to 2 players and each person may choose from a selection of characters at the beginning menu and play through the dungeon.

We went with a classic Zelda / Gauntlet style feel for the game. The characters and enemies move and attack each other in real time. The ultimate goal of the game is for the players to reach the exit in the map alive. The exit is denoted as a water well. If all players die before reaching the exit, the game is lost.

Trees and chests spawn on the map and are collidable and destroyable. This means that trees can get “cut down”. Our original plan was to have an animation for this but it was low priority and we never got to it. So, if you can walk on a tree or a bush that probably means that a Gunter or yourself (or a teammate) killed the tree and it became uncollidable. When you break a chest, it disappears and it spawns the items that were in its inventory. Right now, only a single sword is an item and it affects your speed and attack strength. Whenever a chest is broken you can pick up the sword that it spawns.

Patterns

## State

#### MainMenuState, GamePlayState, PauseState, GameOverState, GameWonState

We used the state pattern to move between different sections of the game, primarily menus and the actual game play. This provides an easy means to change the interface for the user and add more states later.

## Singleton

#### All Manager classes

We decided to use the Singleton pattern on the Managers so that there would be a centralized area to make changes to the game on the fly. With these managers being singletons, it helps to ensure that there are no components in the game out of sync.

## Strategy

#### Attacks and Menus

Strategy provides a common interface for managers to quickly perform actions. Attacks may vary quite a bit in games, so strategy is a natural pattern to use here. We used it in Menu because menus are drawn very similarly with some slight variations that the concrete classes can take care of.

## Flyweight

#### Art Asset Handling

Because of the large number of sprites to be drawn in the map, there could be significant lag based on map size. However, many of these sprites are the same texture. Flyweight allows us to query for a commonly used texture and use a reference to a single sprite instead of bogging the game down with copies.

## Factory

#### Building Entities and Items

Factory provides a quick and easy way to generate random items and entities on the fly. This allows for significantly less hard coding to create varying objects that differ mostly in data.

## Builder

#### Map data generation

The map data is very complex. Procedural generation of a random map may become convoluted and hacky. The builder pattern simplifies this. The algorithms in generation can direct the builder which creates the map, decoupling the algorithms, allowing us to change them out if necessary.

Game Play

# Controls

## Player 1

Movement: WASD Keys

Attack: Space

Pick-up Item: E

## Player 2

Movement: Arrow Keys

Attack: Right-Ctrl

Pick-up Item: Right Shift

## Other

Toggle Debug Display: ~ key

This displays information on the players, and the map (such as the end goal)

The text to the right of the screen will display the Spawn and End points in order to make it easier to traverse through the forest.

The text to the left of the screen will show your current position.

Pause: ESC

Exit: F4

# Player Selection

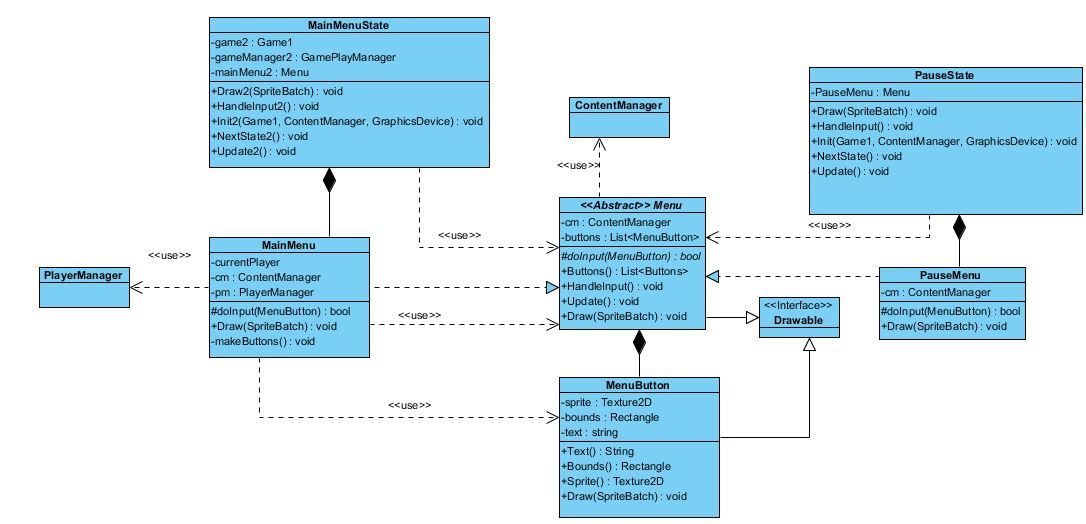
When the game starts, players may choose the characters to use at the menu. Up to 2 players may be in the map at once.

# Goal

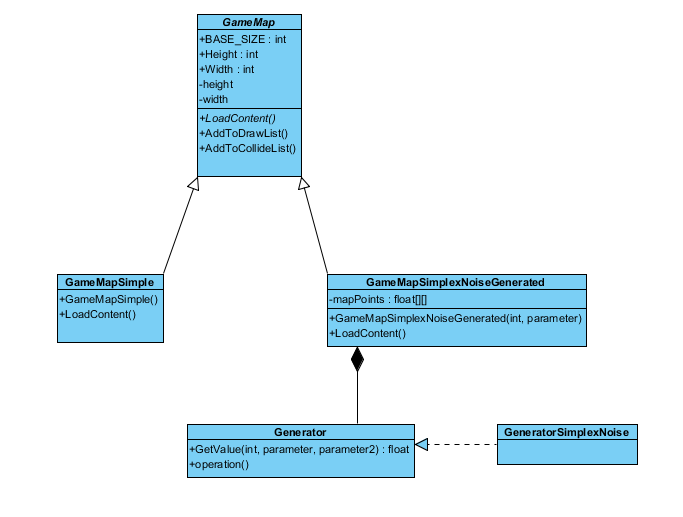
The goal of this game is to reach the end (water well) alive.

# Game1

# Menu



# Map Generation



# Game State

