Andy Herrold

CS 120

Final Project

**Beat the Dealer**

**Game Design Document**

**Andy Herrold**

“Beat the Dealer” will be a simple 2D card game utilizing pygame and simpleGE. It is based upon the popular casino game blackjack. The player will be represented by two random cards at the bottom of the screen. The dealer will be represented by two random cards near the top of the screen. The players objective will be to make a hand with a higher total than that of the dealer without exceeding a total of 21 (“busting”). A casino or blackjack table image will make up the background. Cards will be represented by a card image that corresponds to the random cards assigned to the player and dealer.

When a card is dealt, it will be accompanied by a corresponding card deal sound. When a bust occurs, it will be accompanied by a corresponding popping sound. When a blackjack occurs (first two cards totaling twenty-one) in either the player or dealer hand, a jackpot sound will play.

The game will start with an instruction screen. This screen will display the basic mechanics and objectives of the game. It will contain two buttons: Play and Quit. Play will initiate the play state and Quit will exit the game.

When the game round has finished, the player will be taken back to the Intro screen where their score will be displayed.

**State Transition Diagram**

**A spiral notebook with a diagram

Description automatically generated**

This game uses a standard two-state system. Each state represents a subclass of the simpleGE Scene class. The game begins on the intro screen. This screen will contain the player instructions, the start button, and the quit button. Each button will set a response variable and close the screen base on the players’ choice. The play button will send the player to the game play scene. The quit button will end the game.

The game play scene will always end when the player bankroll reaches zero, and always returns control to the intro scene. However, it does pass back its score to the main function, which uses that score to provide feedback to the user in the intro scene.

**Instructions Scene**

**A notebook with writing on it

Description automatically generated**

Four visual elements:

**Instructions –** SimpleGE multilabel explaining the game play instructions

**prevScore –** stock label showing previous score

**btnPlay –** stock button indicating “Play”

**btnQuit -** stock button indicating "Quit”

Other attributes:

**prevScore** - integer indicating last score, passed into the class initializer and displayed on prevScore label

**response** - string representing the user's intentions. Set by the two buttons and read in the main function

Initializer will create all attributes and set up a sprite list

Init (score)

Set image to las vegas.jpg (Image by Pexels by Pixabay)

Set response to “Play”

Create instructions multiLabel

Add textLines containing instructions

Set instructions center to ()

Set instructions size to ()

Copy score parameter to prevScore attribute

Create LblScore

Set text to “Last score: {prevScore}”

Set center to ()

Create btnPlay

Set text to “Play”

Set center to ()

Create btnQuit

Set text to “Quit”

Set center to ()

Add lblInstructions , lblScore, btnQuit, and btnPlay to sprites

**The Game Class**

Primary class of the game. It will be subclassed from the simpleGE.scene

A notebook with writing on it

Description automatically generated

Game Class Attributes:

**Player Hand -** An instance of the PlayerHand Class (see below)

**Dealer Hand -** An instance of the DealerHand Class (see below)

**lblScore -** An instance of the lblScore class (see below)

#need to figure out “Hit” and “Stand” button

Non-sprite assets:

**Deck** ()

**Score** ()

**SoundDeal ()**

**SoundWin ()**

**SoundBust ()**

**SoundBlackjack ()**

**Initializer will create all needed components:**

Init:

Set image to lasvegas.jpg

Create deck

Random list

Set score to ten

Initialize SoundDeal

Initialize SoundWin

Initialize SoundBust

Initialize SoundBlackjack

Create instance of PlayerHand

Create instance of DealerHand

Create instance of LabelScore

Add PlayerHand, DealerHand, and lblScore to sprites

**All event-handling will occur in the scene’s process () method**

Process:

For each round initialize **deck initDeck ()**

Initialize **SoundDeal**

Assign two cards to player **assignCards ()**

Assign two cards to dealer **assignCards ()**

If  **Playerhand** or **Dealerhand** equal 21:

Play the blackjack sound (soundBlackjack)

If PlayerHand equals 21 add 1.5 to score

If DealerHand equals 21 subtract 1 from score

Reset Deck

Update lblScore

While playerHand is less than 21:

**#create while loop-use keepGoing?**

Display **hitBtn** and **standBtn**

If player selects hitBtn:

Deal one card from deck to playerHand

Play dealSound

If playerHand is > 21:

Play soundBust

Subtract 1 from score

If player selects standBtn:

Deal one card from deck when DealerHand < 17

Deal one card from deck until DealerHand => 17

Else:

Display dealerHand

If dealer hand is between 17 and 21:

Display dealerHand

If dealerHand > playerHand:

Subtract 1 from score

If dealerHand < playerHand:

Play soundWin

Add 1 to score

Update Label Score

**Components of the game class**

**Each visual element of the game class is an extension of a simpleGE element.**

**playerHand**

**????**

**Size should be ?**

**Position should be lower middle portion of screen**

**dealerHand**

**?????**

**Position should be upper middle portion of screen**

**lblScore**

**lblScore is a subclass of the simple.GE.Label**

**Has text and center with no events**

Init ():

Set text to “Score: 10”

Set center to ?

**hitBtn**

**hitBtn is a subclass of the simpleGELabel**

**position should be left middle side of screen**

**stndBtn**

**stndBtn is a subclass of simpleGElabel**

**position should be right middle of screen**

**The main () function**

**The main function will manage the high level state transition between the intro and play states. It is a main loop with four variables**

**Instructions-an instance of the Instructions class**

**Game-an instance of the Game class**

**keepGoing-classic Boolean sentry**

**score-the current score**

main ()

Set keepGoing to True

Set score to ten

While keepGoing is True:

Create an instance of instructions

Pass the current score to instructions as a parameter

Start instructions

When instructions ends:

If instructions.response is “Play”:

Create an instance of Game->game

Start game

When game is over, copy game.score to score

If instructions.response is anything but “Play”:

Set keepGoing to False, which will exit the game

**Milestone Plan**

**Create gameplay followed by instruction scene. Integrate with state management. Store game process on github, with marked commit for each milestone reached and other commits as needed. Each milestone commit will run correctly with the milestone demonstrated. Each milestone is expected to take one programming session to complete.**

1. **Game scene with background image**
2. **Add Quit Button and Stand Button**
3. **Add Score Label**
4. **Configure Deck Database**
5. **Create Player Hand**
6. **Create Deler Hand**
7. **Create Sound effects for win, bust, deck, and blackjack**
8. **Add instruction class and state transition**
9. **Add special blackjack rules (split, double, etc.)**

**Asset Plan:**

**Lasvegas.jpg**

**A cityscape with a tall tower and a hot air balloon

Description automatically generated**

Image by <a href="https://pixabay.com/users/pexels-2286921/?utm\_source=link-attribution&utm\_medium=referral&utm\_campaign=image&utm\_content=1846684">Pexels</a> from <a href="https://pixabay.com//?utm\_source=link-attribution&utm\_medium=referral&utm\_campaign=image&utm\_content=1846684">Pixabay</a>