Homerun!!!

Game Design Document
Taishi Hiraishi

Overview

"Homerun!!!" will be a basic 2D arcade game to demonstrate the overall flow of a game using pygame and simpleGE.

The player is the hitter. Hitter appears to the left of the gameplay screen, and a background image of the baseball field is behind him. The user can move the hitter up and down with the corresponding arrow keys. 5 baseball is thrown from the right side of the screen by the pitcher. Each baseball will be thrown from a different y position and at a different speed between 1 and 15 pixels per frame. If the hitter touches a ball, a positive sound effect is played, and the hitter's image will turn into a celebrating image for a moment, and it gains a hit. If a ball leaves the left of the screen and collides with the transparent catcher, it is reset to a new random position at the top of the screen and a new falling speed. Also, the pitcher gains one strike. If three strikes are taken by a pitcher, that will turn into a single out, and three outs are equal to one inning. The objective of the game is to hit as many hits as you can before the 9th inning ends.

When the game begins, it will show an intro screen with instructions and two buttons. The play button will take the game into the play state. The quit button will exit the game.

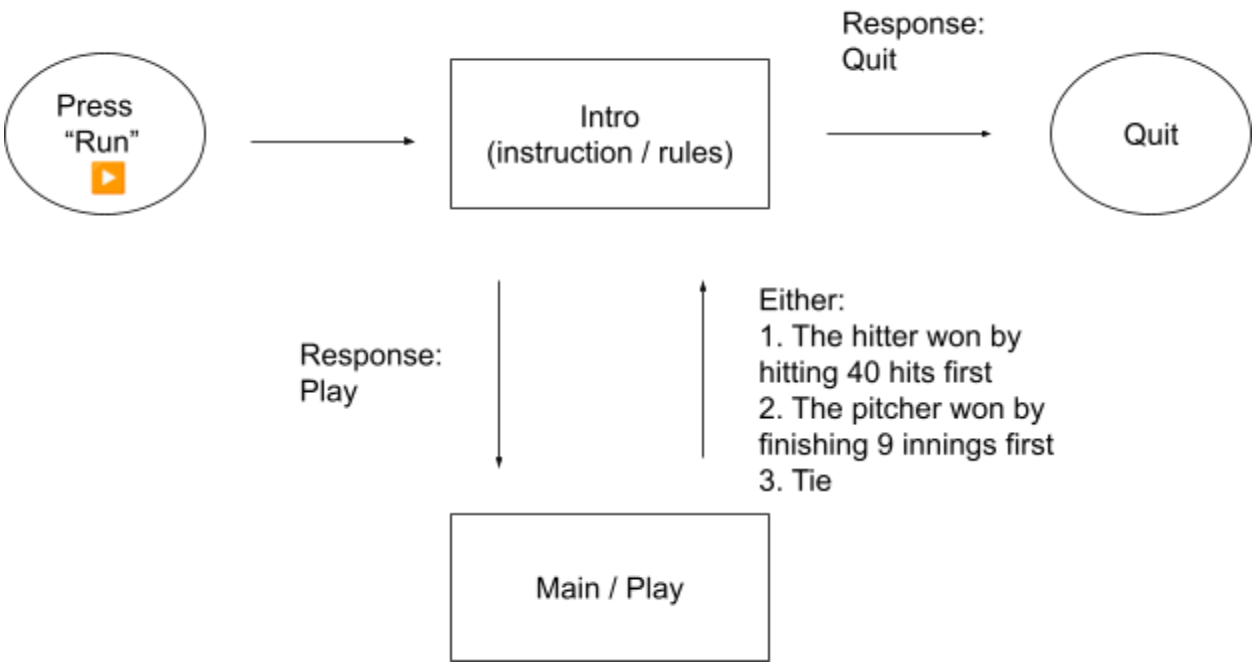
.

State Transition Diagram

The player is initially sent to the Intro scene, which shows instructions & rules, and two buttons that ask them if they want to play the game or quit it. If the user chooses to play the game, she is sent to the gameplay scene. If she chooses to quit, the game ends.

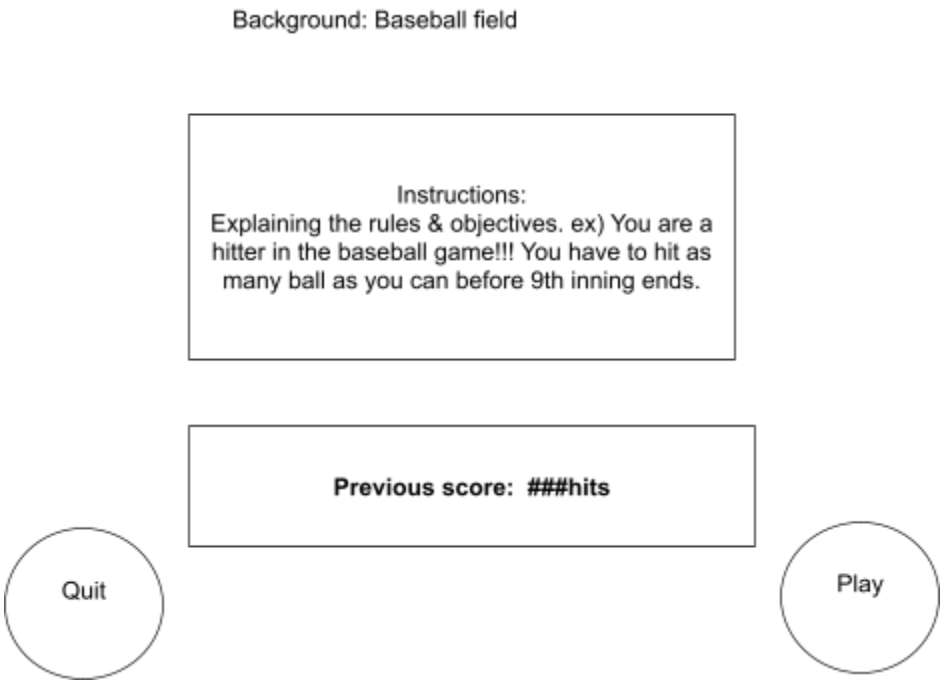
The gameplay scene always ends if a hitter hits 40 hits faster than pitching and vice versa. The players are put back to the intro scene.

When the player comes back from the playing state, it holds the record of numbers of wins by both pitchers and hitters



The Instructions Scene

The instructions scene is simple but vital: It controls access to the game and explains how the game works to the users.



This scene has four main visual elements:

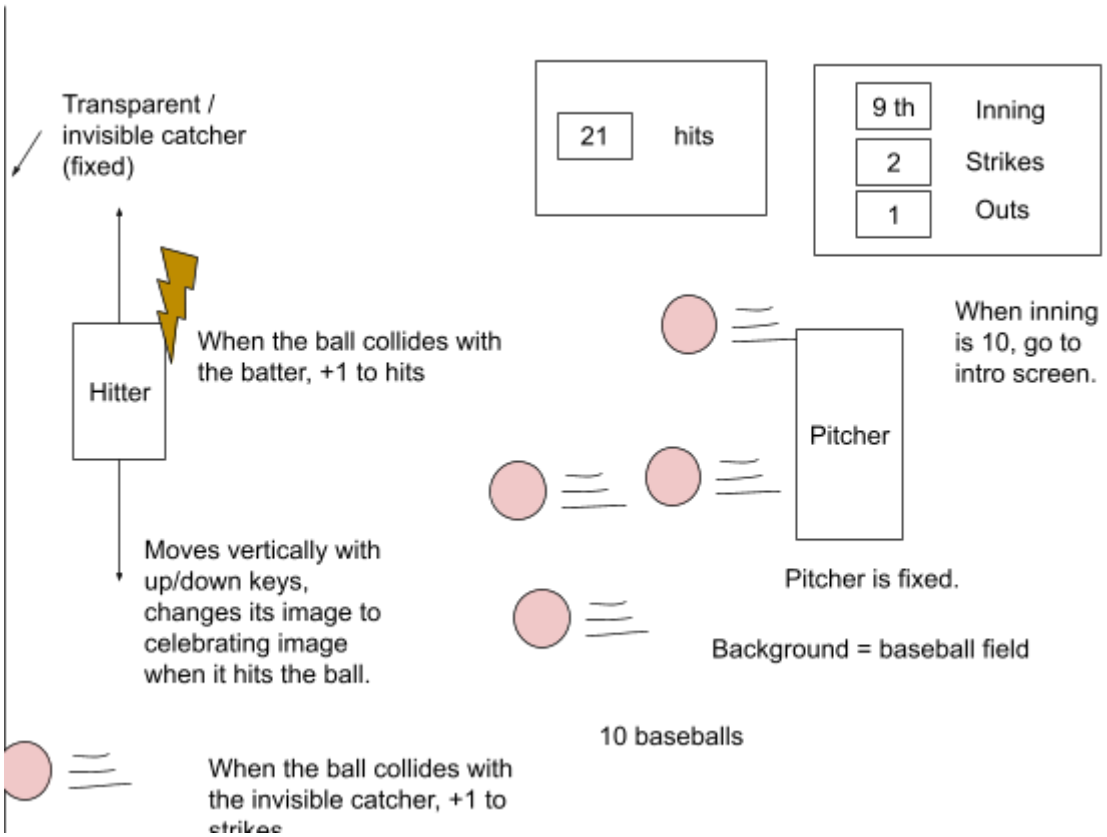
- **instructions** - a stock simple multilabel containing instructions for game play
- **stats** - a stock label showing the records of wins by each side
- **btnPlay** - a stock button indicating "Play"
- **btnQuit** - a stock button indicating "Quit"

Other attributes:

- **stats** - an integer indicating the historical records that keeps the number of wins by each side.
- **response** - a string representing the user's intentions of playing the game or not. “ buttons are used to show play & quit

The Game class

This is the primary class of this game. It will be subclassed from simpleGE.Scene



The Game class will have a number of visual attributes:

- **hitter** - an instance of the **Batter** class (see below)
- **pitcher** - a list of instances of the **Pitcher** class (see below)
- **transparentCatcher -** a list of instances of the **TransparentCatcher** class (see below)
- **ball** - a list of instances of the **Ball** classes (see below)
- **lblInnings** - an instance of the **LblInnings** class (see below)
- **lblOuts** - an instance of the **LblOuts** class (see below)
- **lblStrikes** - an instance of the **LblStrikes** class (see below)
- **lblHits** - an instance of the **LblHits** class (see below)

It will also contain some non-sprite assets:

- **Innings** - an int containing the current innings
- **Outs** - an int containing the current outs
- **Strikes** - an int containing the current strikes
- **Hits** - an int containing the current hits
- **sndHits** - a stock instance of the simpleGE.Sound class
- **sndStrikes** - a stock instance of the simpleGE.Sound1 class
- **sndOuts** - a stock instance of the simpleGE.Sound2 class
- **sndInnings** - a stock instance of the simpleGE.Sound3 class

The initializer will create all the needed components:

```
init:
  Set image to field.jpg
  Set innings,outs,strikes, and hits to zero
  Initialize sndHits to hit sound effect
  Initialize sndStrikes to strikes sound effect
  Initialize sndOuts to outs sound effect
  Initialize sndInnings to innings sound effect
  Create an instance of Hitter/Batter -> hitter/batter
  Create an instance of Pitcher -> pitcher
  Create an instance of TransparentCatcher-> transparentCatcher
  Create list of (5) Balls instances -> balls
  Create instance of LblInnings
  Create instance of LblStrikes
  Create instance of LblOuts
  Create instance of LblHits
```

```
  Add Hitter/Batter, Pitcher, TransparentCatcher, Ball, LblInnings,
LblOuts, LblStrikes, LblHits,to sprites
```

All event-handling will occur in the scene's process() method:

```
process:
  For each ball in the ball list:
     If that ball collides with hitter:
     Play the ball collision sound (sndBall)
     Reset that ball
     Add one to the hit
     Update lblHit to indicate the new score

   For each ball in the ball list:
     If that ball collides with transparentCatcher:
     Play the ball collision sound (sndStrikes)
     Reset that ball
     Add one to the strikes
     Update lblStrikes to the new score
     If strike == 3,
     Play the out sound (sndOuts)
     Add one to the outs
     Strikes = 0
     Update lblOuts to the new score
     If outs == 3,
     Play the out sound (sndinnings)
     Add one to the innings
     outs = 0
     Update lblInnings to the new score
```

Components of the Game class

Each of the visual elements of the Game class is an extension of a simpleGE element.

Hitter

Hitter is a subClass of simpleGE.Sprite
The image is a copy-right free image of a hitter
Size should be roughly 50 by 70
Transparent background is preferred

Initial position center left of screen
Set it so it will move 7 pixels per frame
Set a celebrating image of the hitter after it hit the ball
Set contact duration to 500ms
Set contact timer to 0

```
init:
  Set originalImage to batter.png
  Set size to 50x70
  Set position to (120, 240)
  Set moveSpeed to 7
  Set contactImage to homerun.png
  Set contactDuration to 500
  Set contactTimer to 0
```

All event-handling will be in process() method
Move left on left key, right on right key
Change image to homerun.png when in contact with the ball.

```
process:
  If left key is pressed
     Subtract moveSpeed from x
  If right key is pressed
     Add moveSpeed to x
If contactTimer is more than 0, subtract 1000/30 from the the
timer (5oo)
If contactTimer is less or equal to 0, put the image back to
originlImage.
```

Ball

Ball is a subclass of simpleGE.Sprite
The image should be a copy-right-free image of the baseball
It should have a transparent background
Reset method sets the ball to right of screen, random position
Fall speed is random within limits (-15 to -1 ppf)
The Ball moves on the screen from right to left
If the Ball leaves left of the screen, reset
Ball-Hitter collision handled at game level, not needed here
Ball has no special attributes, but two methods

- **init()** - standard initialization
- **reset()** - custom method to set speed and position

```
init():
```

```
  Set image to ball.png
  Set size to 25x25
  Call reset()

reset():
  Set x to 520
  Set y to random from zero to screen height
  Set dy to random between -15 and -1
```

Pitcher

Pitcher is a subclass of simpleGE.Sprite
The image should be a copy-right-free image of the pitcher
Size should be roughly 50 by 70
It should have a transparent background
Initial position center right of the screen & fixed
Pitcher has no special attributes, and one methods

- **init()** - standard initialization

```
init():
  Set image to Pitcher.png
  Set the size to 50x70
  Set position to (520, 240)
```

TransparentCatcher

Transparentcatcher is a subclass of simpleGE.Sprite
The image should be a copy-right-free image of the catcher
Size should be roughly 1 by 480
It should have a transparent background
Initial position center left of the screen & fixed
TransparentCatcher has no special attributes, and one methods

- **init()** - standard initialization

```
init():
  Set image to Pitcher.png
  Set the size to 1x480
  Set position to (0, 240)
```

LblInnings

LblInnings is a subclass of the simpleGE.Label

It is quite simple - could have been a stock instance
It simply has text and center, no events

```
init():
  Set text to "Innings: 0"
  Set center to (500, 300)
```

LblOuts

LblOuts is also a simple subclass of simpleGE.Label
Again, only text and center, no events

```
init():
  Set text to "Outs: 0"
  Set center to (500, 270)
```

LblStrikes

LblStrikes is also a simple subclass of simpleGE.Label
Again, only text and center, no events

```
init():
  Set text to "Strikes: 0"
  Set center to (500, 240)
```

LblHits

LblHits is also a simple subclass of simpleGE.Label
Again, only text and center, no events

```
init():
  Set text to "Hits: 10"
  Set center to (450, 290)
```

The main() function

The main function will manage the high-level state transition between intro and play states.
It is a very standard main loop, containing four variables:

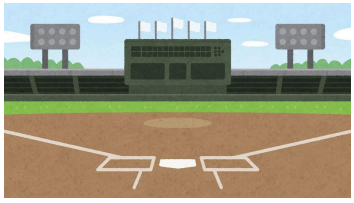
- **instructions** - an instance of the Instructions class
- **game** - an instance of the Game class
- **keepGoing** - the classic Boolean sentry
- **winsP** - the current number of games won by a pitcher
- **winsH** - the current number of games won by a hitter

Psuedocode for main

```
main():
  Set keepGoing to true
  Set winsP,WinsH to zero
  While keepGoing is true:
      Create an instance of Instructions -> instructions
      Pass the current winsP & winsH 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.winsP or WinsH.
      If instructions. response is anything but "Play":
      Set keepGoing to False, which will exit the game
```

Asset plan

field.jpg
Fair use from https://www.irasutoya.com/ , Japanese copy-right-free website



hitter.png
Fair use from https://www.irasutoya.com/ , Japanese copy-right-free website

ball.png
Fair use from https://www.irasutoya.com/ , Japanese copy-right-free website

pitcher.png
Fair use from https://www.irasutoya.com/ , Japanese copy-right-free website

Catcher.png
Fair use from https://www.irasutoya.com/ , Japanese copy-right-free website

Homerun.png
Fair use from https://www.irasutoya.com/ , Japanese copy-right-free website

Hit.wav
Fair use from https://soundeffect-lab.info/, Japanese copy-righy-free website

Strikes.wav
Recorded and made by myself

Outs.wav
Recorded and made by myself

Innings.wav
Recorded and made by myself