**Basketball Hoop Game**

Game Design Document

John McDaniel

Overview

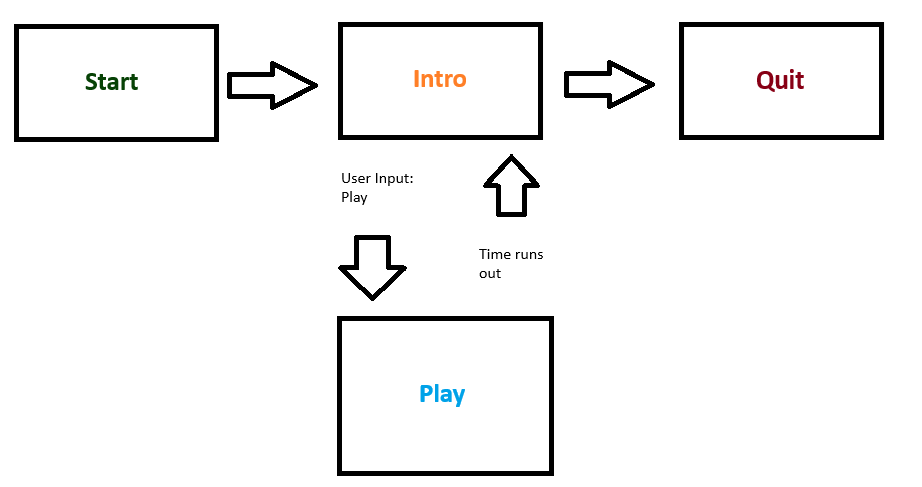
The Basketball Hoop Game will be a simple 2D game that introduces programmers new to Python using pygame and simpleGE.

The overview is simple: the user is a basketball hoop trying to catch as many basketballs as possible in the allotted time. In this case, there will be ten seconds on the clock. The hoop appears at the bottom in front of the basketball court. The user can move the hoop left and right using the arrows on the keyboard. Each basketball will fall 3 to 8 pixels starting at a random place up at the top, making it challenging but not impossible. When a basketball and a hoop collide, a point will be added, a switch noise will be made, and another ball will fall again. If a ball leaves the screen, it will reset at a random position at the top.

When the game starts, it will show a screen with instructions in the middle and two buttons underneath, allowing the player to start or quit.

After the player runs out of time, the game returns to the intro screen, which shows the score and allows the player to play or stop the game.

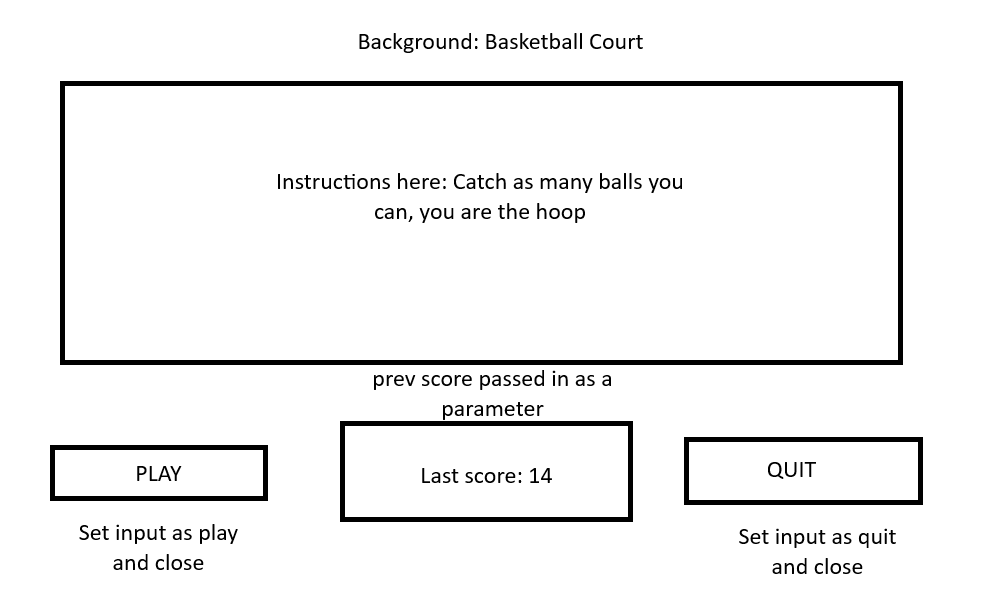
Game Design



The Basketball Hoop Game is a simple system that gives the user two options and instructions when they start the game. Each option is represented as a subclass in the simpleGE Scene class. The buttons close out the current screen but bring up two different screens depending on which button is selected. If the user hits play, then the game will start. If quit is selected, then the tab will be closed. The game screen will end when time runs out, and a screen allowing the user to play again or quit will appear. It will also have the user's previous score on the screen.

The Introduction Screen

The introduction is easy to understand, and the game directions, play button, quit button, and last score are displayed.



The intro has four elements:

1. Instructions- simpleGE multiLabel containing instructions for game play
2. prevScore- label showing the previous score
3. btnPlay- button allowing the user to play
4. btnQuit- button allowing the user to quit

Pseudocode

Class Instruction(simpleGE.Scene):

\_\_init\_\_

Set background

Set response to quit

Create directions label

Create Play and Quit buttons

Create label showing last score

Add directions, btnplay, btnquit, lblScore to sprites

Def process():

If Play button clicked:

Set response to "Play”

Stop scene

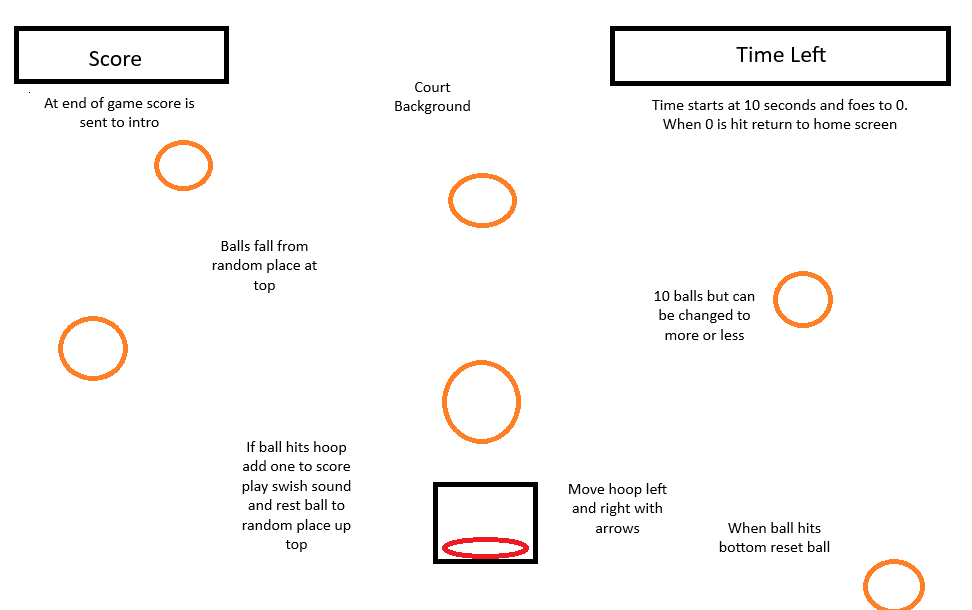
If Quit button clicked:

Set response to "Quit"

Stop scene

The Game Class

Main gameplay scene with timer, player, balls, scoring, and collision using simpleGE subclasses.



The game will have many visuals

1. Balls- balls class below
2. Basketball hoops- basketball class below
3. lblScore- LblScore class below
4. lblTime- LblScore class below

Many non-sprite assets

1. Timer- simpleGE.Timer
2. Score- int containing score
3. sndBall- ball sound class

Pseudocode

Class Game (simpleGE.Scene):

\_\_init\_\_

Set background

Load sound

Set initial score and timer

Create hoop sprite

Create multiple Ball sprites

Create score and time label

Add basketball, balls, lblScore, lblTime to sprites

Def process():

For each ball:

If ball collides with hoop:

Reset ball

Play sound

Increase score

Update score label

Update time label with time left

If timer is finished

Print final score

Stop scene

Ball Class

Represents a basketball falling from the top.

Pseudocode

Class Ball (simpleGE.Sprite):

\_\_init\_\_

Set image and size

Define speed range

Call reset()

Def reset()

Set y to top of screen

Set x to random horizontal position

Set falling speed to random value

Def checkBounds()

If ball goes off bottom:

Reset position

Basketball Class

Player-controlled hoop at the bottom of the screen.

Pseudocode

Class Ball (simpleGE.Sprite):

\_\_init\_\_

Set image, size, and start position

Set movement speed

Function process():

If LEFT key pressed: move left

If RIGHT key pressed: move right

LblScore and LblTime Class

Display current score and time left as labels.

Pseudocode

Class LblScore(simpleGE.Label):

\_\_init\_\_

Set text to "Score: 0"

Set position

Class LblTime(simpleGE.Label):

\_\_init\_\_

Set text to "Time left: 10"

Set position

Main function

The main function will mange the different transition from intro and play using a loop.

1. Instructions- open the instructions class
2. Game- open the game class
3. Keepgoing- Boolean sentry
4. Score- current score

Psuedocode

Function main()

Set keepGoing to True

Initialize lastScore to 0

While keepGoing:

Create and start Instructions scene with lastScore

If player chooses Play:

Create and start Game scene

Update lastScore with game scorw

Else:

Set keepGoing to False

Milestone Plan

General strategy is to create gameplay first, then instructions screen, and finally integrate with state management.

1. Game scene with background image

2. Add basic Charlie sprite

3. Add keyboard motion to Charlie

4. Add single coin with reset, falling and boundary behaviors

5. Add collision effect between charlie and coin, sound effect

6. Modify for multiple (ten) coins including collision behavior

7. Add scorekeeping, timing, and appropriate labels

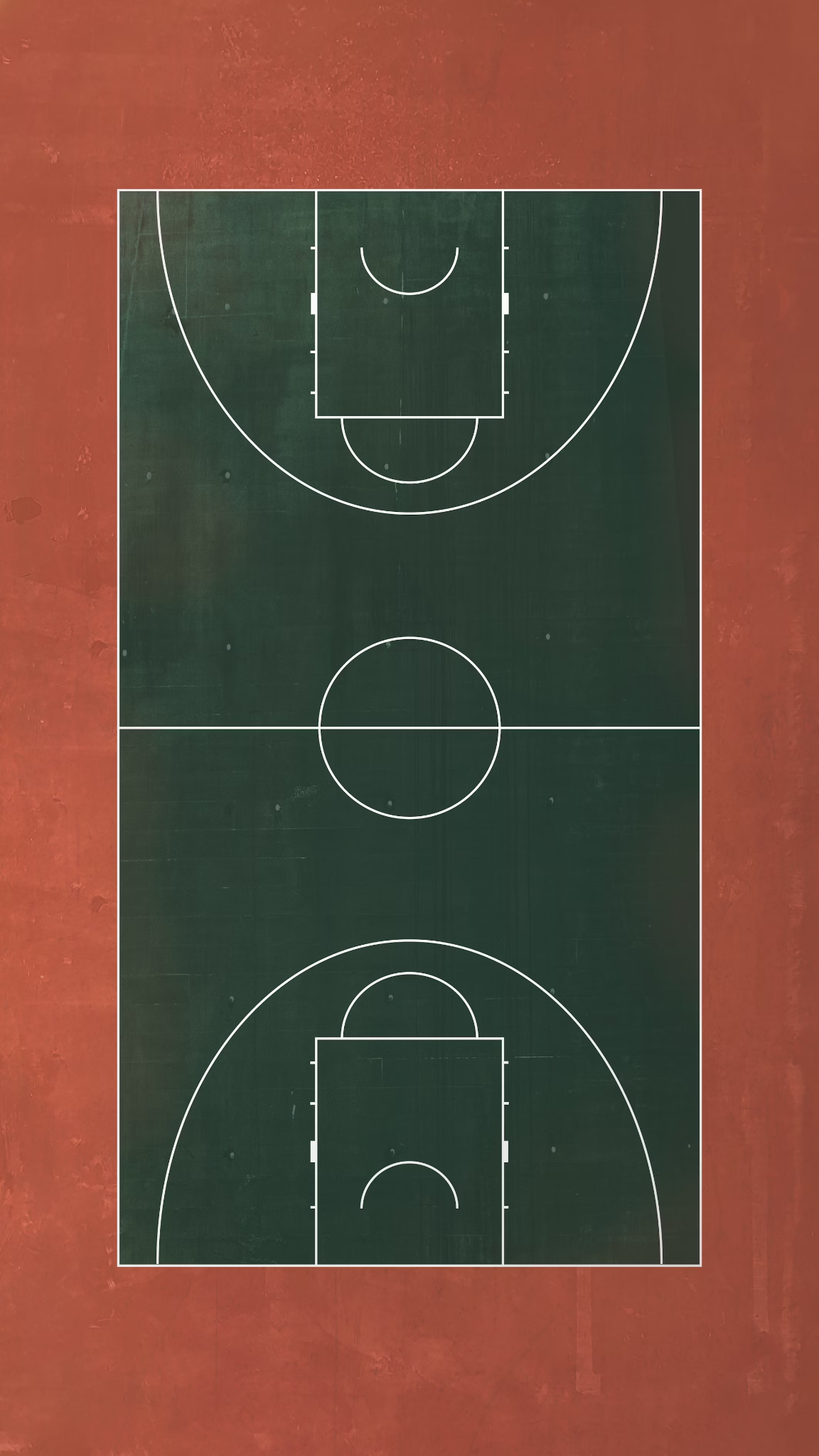
8. Add instructions class and state transition

Assets Plan

Ball1.jpg



Fair use off Unsplash



court2.jpg

Fair use off

Upslash

hoop1.jpg



Fair use off Upslash

Swish.wav

Custom audio by John McDaniel with jsfxr: <https://sfxr.me/#8uy6j3hBvB4bjJ5JHFvRTMmyTTmg1d8k41KLRmCM2rvEjMbj6YCJfaha4Wpy1b12sgjuN6Qz5PkLjQcLteUc4NxPhfsuRKovci8QA13tbBD5P3ivTjKxLQ6yr>