Catch the Cats

Game Design Document

Lucas Lehman

## Overview

"Catch the Cats" will be a simple game to practice using pygame and simpleGE.

The player will control a basket for catching the cats. There will be a background with a tree that the cats are falling from. The user can move the basket to the left and right with the arrow keys. The cats fall from the top of the screen. Each cat will fall from a different x position, and at different speeds, straight down. If the basket touches a cat, a meowing sound is played, and the score is increased. If a cat leaves the bottom of the screen, a hissing sound is played and it is reset to a new random position at the top of the screen at a new falling speed. The game continues for a set period of time.

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.

After the player has played a round of the game, they are taken back to the intro screen. This will display the latest score achieved.

## State Transition Diagram

A diagram of a diagram

Description automatically generated

This game will have two-states. Each is a subclass of the simpleGE Scene. You start on the Intro scene with the instructions and two buttons. The buttons close the scene and set a response variable showing the user's preference. If the user chooses play, they are sent to the game play scene. If they choose quit, the game ends.

The game play scene ends when the time expires, and goes to the intro scene. Before though, it passes it’s score to the main function, which displays that score on the intro scene.

## The Instructions Scene

controls access to the game

A screenshot of a computer game

Description automatically generated

This scene has four visual elements:

* **instructions** - a simpleGE multiLabel with instructions
* **prevScore** - a label showing the previous score
* **btnPlay** – a button reading "Play"
* **btnQuit** - a button reading "Quit"

Other attributes:

* **prevScore** - integer that holds the last score, gets passed into the class initializer and onto the prevScore label
* **response** - string for the user's response. Set by the exit/play buttons and read in the main function

Initializer creates all attributes and sets up the sprites

init(score):

Set image to tree.jpg

Set response to "Play"

Create instructions MultiLabel

Add textLines containing instructions

Set instructions center to (320, 240)

Set instructions size to (500, 250)

Copy score parameter to prevScore attribute

Create LblScore

Set text to "Last score: {prevScore}"

Set center to (320, 400)

Create btnPlay

Set text to "Play"

Set center to (100, 400)

Create btnQuit

Set text to "Quit"

Set center to (550, 400)

Add lblInstructions, lblScore, btnQuit, and btnPlay to sprites

Event-handling happens in the method process()

process():

If the quit button is pressed:

Set response to "Quit"

Stop the scene

If the play button is pressed:

Set the response to "Play"

Stop the scene

## 

## The Game class

The main class of the game. It is a subclass of simpleGE.Scene

A diagram of a cat

Description automatically generated

The Game class has 4 visual attributes

* **Basket**- an instance of the **Basket** class
* **cats** - a list of instances of the **Cats** class
* **lblScore** - an instance of the **LblScore** class
* **lblTime** - an instance of the **LblTime** class

It also has non-sprite assets:

* **timer** – a instance of the simpleGE.Timer class
* **score** - an integer with the score
* **sndCat** – an instance of the simpleGE.Sound class

Initializer creates all the components

init:

Set image to tree.jpg

Create timer

Set timer's total time to 20 (might be changed)

Set score to zero

Initialize sndCat to meow sound effect

Create instance of Basket named basket

Create list of 10 Cat instances named cats

Create instance of LblScore

Create instance of LblTime

Add basket, cats, lblScore, lblTime to sprites

Event-handling happens in the method process()

process:

For each cat in the cats list:

If that cat collides with basket:

Play the cat collision sound (sndCat)

Reset that cat

Add one to the score

Update lblScore to indicate the new score

Update lblTimer with the current time left

If the time left is less than zero:

(for now) print the score to the console

Stop the game

## Components of the Game class

Each visual element of the Game class is based on a simpleGE element.

### Basket

subClass of simpleGE.Sprite

Free use image of basket

Around 100 by 50?

Transparent background

Initial position is the center bottom of screen

moveSpeed attribute is 5 (for now)

init:

Set image to basket.png

Set size to 100x50

Set position to (320, 400)

Set moveSpeed to 5

event-handling will be in process() method

Move left on left key, right on right key

process:

If left key is pressed

Subtract moveSpeed from x

If right key is pressed

Add moveSpeed to x

### Cat

subclass of simpleGE.Sprite

Picture of my cat Murphy

Transparent background

Reset method sets cat to top of screen at a random position

Fall speed is random (with limits)

Cat falls down screen

If cat leaves bottom of screen, reset

Cat has three methods

* **init()** - initialization
* **reset()** - custom method for speed and position
* **checkBounds()** – write over the existing checkBounds to handle bottom-of screen

init():

Set image to cat.png

Set size to 25x25

Call reset()

reset():

Set y to 10

Set x to random from zero to screen width

Set dy to random between 3 and 8

checkBounds():

If bottom of sprite is larger than screen width:

Call reset()

### LblScore

subclass of the simpleGE.Label

has a text and a center

init():

Set text to "Score: 0"

Set center to (100, 30)

### LblTime

subclass of simpleGE.Label

text and center

init():

Set text to "TimeLeft: 20"

Set center to (500, 30)

## The main() function

Manages state transitions between intro and play

Contains 4 variables:

* **instructions** - instance of the Instructions class
* **game** - instance of the Game class
* **keepGoing** – for the while loop
* **score** - the current score

Psuedocode for main:

main():

Set keepGoing to true

Set score to zero

While keepGoing is true:

Create an instance of Instructions -> 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

## Assets

**Tree.png**



**Basket.png**

A brown basket with two handles

Description automatically generated

**Cat.png**

A cat sitting on the floor

Description automatically generated

**meow.wav**