Andy Herrold

CS 120

Slide and Catch

**Taxi Driver**

**Game Design Document**

Andy Herrold

**Overview**

“Taxi Driver” is a simple 2D arcade game utilizing pygame and simpleGE. The player acts as the taxi driver which is represented by the taxi image near the bottom of the screen. A cityscape image makes up the background. The player can maneuver the taxi to the left or right utilizing the corresponding arrow keys. Molotov cocktails will fall from the top of the screen. These objects will fall from a different random x position and at different frequencies between 3 and 8 pixels per frame straight down. When the taxi is stuck by a Molotov cocktail, an explosive sound is played, and the players score is decreased. When a Molotov cocktail reaches the bottom of the screen, it resets to a new random position at the top of the screen and takes on a new falling speed. The game continues for 30 seconds.

The game will start with an instruction screen displaying two buttons: Star and Quit. The play button will initiate the play state, and the quit button 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 always ends when the time expires, 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 spiral notebook with writing on it

Description automatically generated**

Four main visual elements:

**Instructions-**stock simpleGE multilabel containing game play instructions

**prevScore-**stock label showing the 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 cityscape.jpg

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 (320, 400)

Create btnPlay

Set text to “Play”

Set center to (100, 400)

Create btnQuit

Set text to “Quit”

Set center to (500, 400)

Add lblInstructions , lblScore, btnQuit, and btnPlay to sprites

**All event-handling will happen in the scene's process() method**

Process ():

If quit button is selected:

Set response to “Quit”

Stop the scene

If Play button is selected

Set the response to “Play”

Stop the scene

**The Game Class**

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

A spiral notebook with writing on it

Description automatically generated

Game class attributes:

**Taxi-**an instance of the Taxi class (see below)

**Molotov Cocktails-**an instance of the Molly class (see below)

**lblScore-**an instance of the LblScore class (see below)

**lblTime-**an instance of the LblTime class (see below)

Non-sprite assets

**Timer(work on next lab)**

**Score(work on next lab)**

**SndMolly(work on next lab)**

**Initializer will create all the needed components:**

Init:

Set image to cityscape.png

Create timer

Set timers total time to 30

Set score to zero

Initialize sndMolly to Molly sound effect

Create instance of Taxi – taxi

Create list of (10) Molly instances – molly

Create instance of LblScore

Create instance of LblTime

Add taxi, molly, lblScore, lblTime, to sprites

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

process:

For each molly in the molly list:

If that molly collides with taxi:

Play the molly collision sound (sndMolly)

Reset that molly

Subtract one from the score

Update lblScore to indicate new score

Update lblTimer with the current time left

If the time left is less than zero:

Print the score to the console

Stop the game

**Components of the game class**

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

**Taxi**

**Taxi is a subClass of simpleGE.sprite**

**Taxi is a fair-use image of a taxi**

**Size should be roughly?**

**The initial position should be at center bottom of the screen**

**moveSpeed attribute is an integer that starts at ?**

init:

Set image to taxi.png

Set size to (?)

Set position to (?)

Set moveSpeed to (?)

**All 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

**Molly**

**Molly is a subclass of simpleGE.Sprite**

**Image should be an encumbrance-free image of a Molitov cocktail**

**It should have a transparent back round**

**Reset method sets molly to top of screen, random position**

**Fall speed is random within limits ()**

**Molly falls down the screen**

**If molly passes the bottom of screen, reset**

**Molly-taxi collision handled at game level**

**Molly has no special attributes, but 3 methods**

**Init() – standard initialization**

**Reset() – custom method to set speed and position**

**checkBounds() – overwrite existing checkBounds to handle bottom of screen**

init ():

set image to molly.png

Set size to 25 x 25

Call reset ()

Reset ():

Set y to 10

Set x to random between zero and screen width

Set dy to randon between 3 and 8

checkBounds ():

If bottom of sprite is larger than screen width

Call reset ()

**lblScore**

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

**Has text and center with no events**

Init ():

Set text to “Score: 0”

Set center to (320, 400)

**lblTime**

**lblTime is a subclass of simpleGE.Label**

Init():

Set text to “TimeLeft: 10”

Set center to (500, 30)

**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 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

**Notes on the main loop**

**For technical reasons, its best to recreate each of the scenes on each pass**

**Each scenes start () method is *blocking***

**Which means the next line of main will not occur until the scene is closed**

**Milestone Plan**

**Create gameplay first, then the instruction screen. Then 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 basic Taxi sprite**
3. **Add keyboard portion to Taxi**
4. **Add single molly with reset, falling and boundary behaviors**
5. **Add collision effect between Taxi and Molly, sound effect**
6. **Modify for multiple (10) molly including collision behavior**
7. **Add scorekeeping, timing, and appropriate labels**
8. **Add instructions class and state transition**

**Asset Plan**

**Cityscape.png**

**A black background with a black square

Description automatically generated with medium confidence**

**Creative commons:**

[**https://openclipart.org/search/?query=cityscape**](https://openclipart.org/search/?query=cityscape)

**taxi.png**

**A yellow taxi with white text on the side

Description automatically generated**

**Creative commons:**

[**https://openclipart.org/search/?query=taxi**](https://openclipart.org/search/?query=taxi)

**molly.png**

**A cartoon of a bottle holding a match

Description automatically generated**

**Creative Commons:**

**https://openclipart.org/search/?query=Molotov+cocktail**