CTD 1D Group E Math Invaders

Table of Contents

Members

- Gangesh Kumar - 1007181
- Haresh Jayant - 1007148
- Julian How - 1006913
- Yu Ying - 1007218
- Sushmitha - 1007057

Overview

In the retro Japanese shooting game Space Invaders, the player is tasked with shooting aliens that slowly approach from space. For our Computational Thinking for Design project, we have recreated the foundational mechanics of this game but translated the aliens and the player's objective to that of solving maths equations. The aliens have been replaced by a set of integers that represent answers to a set of equations that appear on the screen, and only by clearing all equations can the player win the game.

Motivation and Target Users

We hope that this alteration of the game's core mechanics will serve as an educational opportunity for young children in the early stages of school. This includes students in primary school but can also include any other persons wishing to improve upon their mental calculation skills. By providing an engaging medium

in which to practice these skills, students can better motivate themselves to train and practice on these fundamental mathematical concepts.

Gameplay

The game is started by selecting a difficulty on the main menu. Player input in the game is controlled by the left and right arrow keys, and bullets are spawned with the spacebar. 10 images of digits will appear on turtle screen. These are the answers for the 10 questions Players have to answer each question correctly by shooting one of the 10 digits If answer is wrong, a cross will be shown and the player will be asked to try again. For every correct answer, score +10, every wrong answer score -5 All questions must be answered before the numbers can touch the player sprite.

Features

The game generates random math equations that are displayed at the top of the game based on the chosen difficulty level. The numbers that replicate the 'alien invaders' from the original game have identical movement patterns to simulate a sense of urgency. There is a score leader board that showcases the top 5 scores for each difficulty from other players to promote competitiveness. The game also features sound effects and background music to boost engagement.

To run:

install public-pixel-font

```
python3 main.py
```

Controls

`Left` - Move Left
`Right` Move Right
`Space` - Fire

Documentation

Libraries

```
import turtle
import math
import random
import time
import json
from itertools import cycle
import sys
import subprocess
from threading import Thread
```

Directory Structure

```
├── data.json
├── digits
├── functions
│   ├── equations.py
│   ├── game_bg.py
│   ├── game_icons.py
│   ├── jsonfunc.py
│   └── start_page.py
├── game_gifs
├── game_music
├── main.py
└── public-pixel-font
```

- data.json contains the respective user name and their scores
- digits contains the images 0-99 which will be used as 'enemies' in the game
- functions is a folder containing the following python files with the various functions that will be called in main.py

```
#imports function used to send and get data from json file
from functions import jsonfunc.py
#imports function to setup startpage
from functions import start_page.py
#function to create equations for beginner/advanced mode
from functions import equations.py
#imports functions related to background of game
from functions import game_bg.py
#imports functions related to icons used in game eg. player
from functions import game_icons.py
```

Game Music

All background music in this code is played using thread and subprocess with the following syntax

```
if sys.platform == 'linux2':
        bgmusic = subprocess.Popen(["afplay","<soundfile>"])
    elif sys.platform == 'darwin':
        bgmusic = subprocess.Popen(["afplay","<soundfile>"])
```

The above code is then attached to a thread using a function to allow for it to be run concurrently with the game.

Functions

Contains various functions that will be called in main.py

jsonfunc.py

```
#returns sorted scores for advanced and beginner mode to display on
startpage
startpagescrs()
#takes in player name and score and stores it in data.json
endpagescr(nameInput,game_mode,score)
```

start_page.py

```
#Takes in sorted scores and displays top 5 for each mode on the start
screen of game followed by respective icons. together with start page
background and icons as turtle objects. Background music for start screen
is played here
start_page(beginnerScores,advScores)
```

equations.py

```
#generates beginner level questions eg. a + 9 = 13
generate_qn_beginner()
#generates advanced level questions eg. a x 93 = 4836
generate_qn_advanced()
```

game_bg.py

```
#sets up all game icons and 'enemies'(digits)
bg_setup()
#clears/hides background icons
bg_end()
#closes turtle window
quit_game()
```

```
#takes in qn num and eqns dictionary andwrites qn onto turtle screen
write_qn(qn_num,eqns)
```

game_icons.py

```
#setup player,bullet,tick,cross,reset,quit icons
icons_setup()
#hiding icons when game is over
icons_end()
#for bullet to appear and give a sound effect
fire_bullet()
```

main.py

imports and combines functions from above together

```
#all files from function folder are imported so that they can be called in
main
from functions import jsonfunc, start_page, game_bg, equations, game_icons
```

Game Methods

All game methods are encompassed under 3 while loops. 2 is to assist with the reset of the game, the other is to assist with the continuous occurence of the game Dictionary of 10 eqns:ans and 10 turtle:ans is created

How position of answers on screen is determined:

```
# runs a loop making sure no 2 digits will overlap by 'splitting' screen
and then assigning position to digit
x = random.randint(-200+(phase*50),-185+(phase*50) )
y = random.randint(100, 250)
enemy.setposition(x, y)
phase = phase+1
```

How player moves:

```
#when either key is clicked, the x value of the player's coordinates is
edited to 'move' the player icon
turtle.onkey(game_icons.plyr_left, "Left")
turtle.onkey(game_icons.plyr_right, "Right")
```

How bullet is fired:

```
#when space is clicked bulletstate is changed to fire
turtle.onkey(game_icons.fire_bullet, "space")
#when bullet state is changed to fire, we check if there are any digits
with similar xcoordinates to player on screen
if math.isclose(xbull,xcoord,rel_tol=0.15,abs_tol=15):
#if there is then we send the bullet to the digit and run the
collision_enemy_bullet(enemy) enemy is turtle object that bullet collided
with
bullet.goto(xbull,enemy.ycor())
```

What happens when bullet collides with enemy(digit):

```
collision_enemy_bullet(enemy)
#check whether ans assigned to enemy on dic is same as ans assigned to eqn
in dic
#if same 'tick' image is shown tgt with sound effect and user proceeds to
next question
#else user tries again and bullet is reset
```

What happens when player collides with enemy(digit):

```
#checks if any 'enemy' icon hits player icon
isCollision_enemy_player(t1, t2)
Game_Over = True
```

What happens when Game_Over = True:

```
#call functions from game_bg and game_icons to clear screen
#adds in end screen together with quit and reset buttons
#writes player scores to json file
game_pass()
```