**Final Assignment 2022 v1.0**

**Class # 7**

**PRISONER ESCAPERZ**

*The Project Final Report is intended to concisely summarize the outcomes of a project and is the final document in the CCS Project Management Methodology. A Project Final Report is used to document project successes, lessons learned and performance in order to signal improvement in project delivery for the future. This template outlines the content and format of final reports to be used for all information systems projects. The* [*Project Management Office*](http://www.carleton.ca/ccs/project-office) *is your resource for completing this document.*

*A Project Final Report reflects the formal and informal feedback collected from project stakeholders and participants throughout the project. Commonly, a Project Post-Mortem is held to explore the experiences of the participants of the project in more detail. The information collected in this way should align with the details included in the Project Final Report.*

*The Project Final Report is to be developed between the project manager and the project director and should be circulated to project stakeholders and participants for feedback. Finally, the project sponsor should sign off on the details of the Project Final Report before it is forwarded to the Project Management Office for archiving.*

*The project sponsor is responsible for presenting the Project Final Report at a meeting of the Information Systems Steering Committee.*

|  |  |
| --- | --- |
| **University** | Curtin University, WA |
| **Description** | A list of functions used to create simulations of different modes |
| **Unit** | COMP1005 Fundamentals of Programming |
| **Author** | Jaswant Pendem [21026922] |
| **Date** | Semester 1, 2022 v1.0 |

Python 2.7, PyGame 2.1.2

1. **Overview** 
   1. The objective of the simulation is to leave one prisoner to remain trapped inside the prison cell while allowing other prisoners to escape successfully out of the gates without being spotted. You have the control to open and close the gates to enable prisoners to escape out of their cells onto the courtyard. But this will not be easy, as to finish the game, all the prisoners will need to escape out of the main gates which will always be open, while having a specific prisoner trapped inside the cell. The last prisoner behind the bar is called the sacrifice, who is left behind to reduce suspicion from the security. In this case, I have decided to name the game ‘***Prisoner Escaperz’***. The name of the game will help potential players to recognize the objective of the simulation, which is to escape the Prison while leaving one prisoner behind the bars. Likewise, it will be recognized as it is an original simulation, based on a real-life prison based in Northern China called ‘*QinCheng Prison*’, which has real life armies of police geese and dogs who are aiding the community to maximize prison security. Therefore, the prisoners will be forced on to facing infuriated police goose, highly trained police German shepherds and worst of all, facing their own fears. The simulation was made to manipulate the different possibilities for prisoners to escape without being caught, this is highly recommended for gamers, problem solvers, security and emergency operation or services groups etc. To increase the chance of escaping, prisoners must pick up a power up or what we call ‘Steroids’ to dramatically boost their speed, this will be in the prison cell at the start of the simulation. Also note prisoners can only escape out to the courtyard during break time or what we call ‘yard time’ which will only be alerted when the alarm rings. During this time, prisoners can go outdoors and socialize with their mates however if one touches the border, the prisoner will be shocked simultaneously a beacon will be lit to signal the security team of a possible escapee.
2. **User Guide - Guide to using the program**

**2.1 Extensions**

1. Movement: The Prisoners will move in a random manner and can increase in speed if a power-up is used. The police goose will be guarding the closed gates by moving horizontally and once the gate is open (it will disappear) the police goose will move to the corner. Police dogs are quick in a L -shape repetitively.
2. Boundaries and Terrain: The inner border/wall is the first boundary represents the prisoner cell which the prisoners will be trapped in, this is because the walls are high, however this can’t be seen as the simulation will be played in birds eye view. The outer boundary will have two exits which is where the prisoners will need to escape through.
3. Interaction: If any police team interact with the prisoners, the prisoners move in the opposite direction or be sent directly back to the prison cell. Note all characters are always active.
4. Non-Moving/Moving Objects:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Features** | **Appearance** | **Size of Blocks** | **Number of Items** | **Ability** |
| **Non- Moving Objects (Stationary)** | | | | |
| Power Up called ‘Steroids’ | Green Blocks | 1 unit | 2 | Increase Speed |
| Alarm or Bell | Red Block in the center | 1 unit | 1 | Rings when gates open = yard time |
| Beacon | Brown Blocks | 3 units | 2 | Lit when someone touches border and shocks one |
| Background | Cream Screen | ∞ | - | - |
| Border 1 | Dark brown blocks | 25 x 35 units | - | - |
| Inner Border | Dark brown blocks | 15 x 25 Units | - | - |
| **Moving Objects** | | | | |
| Police Dog | Orange block | 1 | 2 | Detect prisoners |
| Police Goose | Blue block | 1 | 2 | Open Gate |
| Gate | Brown Blocks | 5 | 2 | Disappears Once Open |
| Prisoners | Yellow/Orange Block | 1 | 8 | Move in random motion |

1. Visualisation/Results: When police dog or goose interact with prisoners’, messages will be printed that “Prisoner x is brutally beaten by Police x “, likewise when a prisoner interact with the wall it will print “Prisoner x has been brutally shocked!”. When a prisoner successfully exits out of the gates it will print “Prisoner x has won!”. All the messages will be printed on the terminal window. I have used different colours and images to differentiate between the characters. I have used a contrasting background colour to make the elements/features to stand out.

**2.2 Using the Simulation**

To start the simulation, you must first input the following steps in the terminal window:

|  |  |  |
| --- | --- | --- |
| **Step** | **Function** | **What it does** |
| **1** | pip show pygame | Checks if your pygame version is version 2.1.2 if not proceed to the next step. |
| **2** | pip3 install pygame --upgrade | To upgrade to pygame 2.1.2 |
| **3** | python3 FOP-Assignment.py | To run the program |

|  |  |
| --- | --- |
| **RULES** | |
| **Keydowns** | **Simulations** |
| Space Bar | To continuously play the simulation |
| Right Arrow Key | Play each frame of the simulation |
| Key D | To toggle debug view or to remove the grid lines |
| Key E | To toggle Evacuation state or open the gates |

Once the simulation loads up, I have inputted a small list of instructions on the bottom left corner of the simulation to assist users with the rules to start the stimulation. The simulation is made to be functionable, accessible and usable by all users by performing Keydown events to allow keypress detection of certain keys.

**Step 4)** It is recommended to start the simulation by clicking on ‘Key D’ to remove the grid lines which might be distracting to viewers when running the simulation.

**Graphical user interface, text, application, email

Description automatically generatedStep 5)** First type in ‘Random Number Generator’ on google and set the parameter between 0 – 7 as there are 8 prisoners in total. The random number generator will randomly choose the number of the prisoner who will remain last also known as the sacrifice.

Figure .“Random Number Generator” of generating numbers between parameters, above is between 0 – 7 as there are 8 prisoners in the simulation. In this case 5 was chosen, therefore the last prisoner remaining in the cell must be prisoner number 5.

Diagram

Description automatically generated with low confidence**Step 6)** Then by clicking the space bar you can visualize the movements of the different sprites or characters in terms of speed and uncontrolled movements. The movement of the sprites can be rapid, those who have picked up power supply, the number of the prisoner will turn from white to green, which will make the prisoner faster.

Figure . Screenshot of the Prisoner Escaperz simulation, only one prisoner must remain, in this case the yellow block shows prisoner No.5 is the last one remaining.

**Step 7)** To pause the movements of the sprites, re-click the spacebar. To play the simulation by each frame click the right arrow key, this will allow you to visualize the movement of each sprite more clearly.

**Step 8)** To win the game in this case, the last prisoner inside the Prison must be Prisoner No.5 to win the game.

1. **Traceability Matrix**

To access the traceability matrix, click [**here**](file:///C:\Users\jaswa\OneDrive\Desktop\Final%20Traceability%20Matrix.xlsx)**.**

1. **Introduction**

The project involves in developing a Python-based code by investigating the different types of parameters and modifications of features to create the backbone of the simulation. To do this I have inputted several methods such as primitive data structures, non-primitive data structures, surface functions, class methods, static method, control flow statements, constructors, operators etc. to create my simulation. The simulation involves at least three different modes which is the power up state, evacuation state, electrocuted state, and simple interaction state, which will be furthered explained in the methodology. To make the output of the simulation I first understood the background of the three different modes for the simulation. I commenced by inputting constructors such as ‘\_\_init\_\_ ()’ functions to initialize attributes of objects by using a parameter list. There can be as many parameters as possible listed, to create instance attributes I have used repetitively used self-variables which will assign the attributes such as the behaviour and properties of an object, nonetheless, to visualise the appearance of the object a class method is first created. When creating a child class method, ‘pass’ function was used to avoid error when no code was allowed. Typically, the functions are listed under the parent class methods.

While setting several parameters I used the non-primitive data structure function called the tuples () function, to *‘stores multiple items in a single variable’* which eases and neatens up the code. Furthermore, when setting the parameters of the objects, I have predominantly used control flow statements such as looping like for loops, while loops and exception handling like if/if not/else/elif statements and branching such as continue and return. Under these statements several a primitive data structure called the Boolean conditional expressions were used; true and false statements utilised to set different types of conditions like a character’s interaction, properties, or behaviours. Likewise, string operations were used with the aid of operators ( ‘+’). For loops are predominantly used through iterable item, like a list in order to perform the same action for each object. For example, all the prisoners have equal attributes and have equal amount of chance to escape as they are all replicas of each other.

I have imported Pygame methods and modified to my own likings to satisfy game requirements.

**Important Pygame Methods:**

* Keyboard Functionality: I was able to manage Pygame’s event subsystem by setting up the keyboard parameter. Since the built-in enumeration of the keys when key-pressed will direct the out-put file to prepare the functions in the input file and once key-released will perform the code.

Text

Description automatically generated

Figure 3. Opened on text editor. The keyboard colour, message and functionality are all defined and to be displayed on the screen as instructions.

* Surface functionality: Surface function such as blit() were used, which are inputted to accept the surface or string as its image parameter, in terms of the position of the new image object at point (x, y). Since the image was a string the image loaded from the images directory (Sprites) by using the blit() function.

A screenshot of a computer

Description automatically generated with low confidence

Figure 4. blit() function used to impact the surface of the background or the display of the simulation.

* Rect functionality: Pygame ‘rect’ was implemented to store , display and manipulate rectangular areas which were used for each object even including the layout of the prison.
* Collision Detection functionality: When two or more objects are in the same coordinates or nearby each other at a particular frame such as an interaction, intersection, or wall collision. Depending on the condition set the object will operate to avoid the object opposite to it. For example, a prisoner will move in the opposite direction to the police team. The prisoner will move with equal momentum when colliding with wall; the conservation of momentum is endorsed.

Text

Description automatically generated

Figure 5. A list of control flow statements is enumerated to define the interactions between objects

When all the code is added the user has the functionality to start the game due to the implementation of keypress functions which will continuously move the prisoners, the police geese guarding the gates and the police dogs moving around the base. The instruction manual is broadcasted on the bottom left screen of the simulation to assist users.

Furthermore, I have imported the police\_goose.png image from the Sprite directory to appear on the taskbar when the simulation is open to create the real-life app environment by showing the users the program is currently running. To do this I have executed:

Text

Description automatically generated

Figure . Code for a police goose portable network graphic as the icon to represent the simulation on the taskbar.

**Static Method:**

Static method is a function decorator was used in ‘pawn.py’ directory, creates utility functions. Performs similar to class method however the methods are bound to a class instead of the object of the class. There are no parameters required, however the class state cannot be modified.

1. **Methodology**

Diagram

Description automatically generatedDiagram

Description automatically generatedOn Python VMware Horizon, the main file loaded up is called ‘FOP-Assignment\_21026922.py’ within there are directories such as ‘FOP-Assignment.py’, ‘Data’ and ‘README’. The Data directory includes the code, sprite and text directories. The Sprite directory is a sub-input directory which is composed of downloaded images to produce the appearance of objects and characters, likewise the text folder is a sub-input directory which includes a downloaded font called ‘Charybdis’ which will be visible when the simulation runs. Finally, the code directory includes the main input directory which will be imported to the output directory called ‘FOP-Assignment.py’. However, all the directories were created automatically as I utilised PyCharm IDE; that way I didn’t have to enter any commands to create the files and directories, instead aided to save time. Above highlighted in the orange are directories generated automatically which includes bytecode cache files and python package directories which makes the program to run little more smoothly and faster.

When loading the output file, using vim ‘directory name’ to open up the text editor. At the very top or start the import function/s are stated to allow importation of modules from the ‘data’ directory (input file) to the main Python Program or ‘FOP-Assignment.py’ (output file).

**NOTE\*:** Before organising the directories, I commenced the project by planning out the appearance by sketching and anticipating the output on OneNote and noted the storyline, characters, movements, and scenarios of the simulation.

Figure . Sketch of the final simulation on OneNote for Windows 10

* In the first mode of the simulation, the prisoners randomly appear in different location inside the prison cell and move at constant velocity in random directions, eventually the prisoner will collect a non-moving object which is the green power up or called the ‘steroid’, which will increase the speed of the prisoner.
* In the second mode of the simulation is the evacuation state, when the gates are open, the alarm will ring to alert the prisoners the opportunity to go outside to the yard
* The third mode of the simulation takes place during yard time, which is when the prisoners touch the outer wall, they will be electrocuted, and this will be known as the beacons will be lit to red.
* Finally in the fourth mode of the simulation are the interactions between the objects. When any member of the security team such as police dogs or police geese interacts with a prisoner, the prisoner will move in the opposite direction to avoid.

Cartoon physics was partially encouraged in the simulation, for example when the gates are opened it is disappeared instead of moving vertically. The movement of the moving-object are not realistic, since they’re moving at a constant velocity rather than taking breaks like stopping nor instantly accelerating.

The idea of the simulation was trying to replicate the simulation as a real-life security camera in bird’s-eye view or drone(aerial) view to maximise security in the prison.

1. **Results (12 marks)**

In my simulation I have 4 different scenarios/modes that will be evident, which are:

* A picture containing timeline

  Description automatically generatedA screenshot of a game

  Description automatically generated with medium confidenceA picture containing qr code

  Description automatically generated**Second Mode:** Evacuation phase, is when the gates open, alarm rings

Figure 9. ‘D-Key’ is clicked. Removes debug view, debug coordinate not viewable. Police goose guards gate. ‘Space bar’ clicked to start movement of moving objects.

Figure 8. Start of the simulation, gridlines to show debug view. Hovering with mouse, will show debug coordinates.

Figure 10. ‘E-Key’ is clicked. Gates are Open which is Yard time or called the evacuation phase for the prisoners. Police goose moves to the corner to spectate any suspicious actions.

* Diagram

  Description automatically generated with medium confidenceDiagram

  Description automatically generated with low confidenceA picture containing diagram

  Description automatically generated**First Mode:** Power Up phase -Picking up power supply to increase in speed

Figure 13. Power Up or Steroid is consumed by 4 before 7. Prisoner 5 and 3 is out of the cell.

Figure 11. Prisoner 4 and 7 approaches power up, whilst prisoner 5 and 3 are going out for yard time.

Figure 12. Prisoner 4 is 1-unit away. Police dog is approaching Prisoner 3 and Police dog approaching Prisoner 5.

* A picture containing diagram

  Description automatically generatedA picture containing diagram

  Description automatically generated**Third Mode:** Electrocuted Phase - Borders are touched the Beacon will be lit, if not touched will remain unlitA picture containing diagram

  Description automatically generated

Figure 16. Prisoner 5 makes a run, whilst signalling beacons to be lit by turning red.

Figure 15. Prisoner 3 moves in opposite direction which is the fourth mode. Prisoner 4 moves quicker. Prisoner 5 recovering from shock moves out of the way and tries to move to the exit.

Figure 14. Prisoner 5 touches the wall and tries to jump the walls, however, is electrocuted. Target is aimed at Prisoner 5. While Prisoner 3 moves is in close contact with the police dog.

* Diagram

  Description automatically generated with low confidence**Fourth Mode:** Interaction Phase - Prisoner Interaction with the Police Dog

Figure 18. Prisoner 3 moves in opposite direction and makes a run.

A picture containing diagram

Description automatically generated

Figure 17. Prisoner 3 in close contact with police dog.

1. **Conclusion/Reflection**

The purpose of the project was to self-program a functionable simulation with three different modes and to gain experience by proactively planning, writing, debugging, documenting, and performing simple operations. An original simulation called ‘Priosner Escaperz’ was made which includes at least three different modes the power up state, evacuation state, electrocuted state, and simple interaction state. The simulation can be considered as an action-maze chase simulation which is similar to Pacman, but in my opinion better.

Overall making the simulation was very tedious and time consuming as using a new software such as PYCharm IDE in the beginning then switching back to Python 3 VMware Horizon as well as learning Pygame were all beneficial at the end of the day.

**My feedback for the simulation**:

* Add music: To upgrade the simulation and to reach to a wider audience, I can add background music and sound effects by using pygame.mixer.init(). Also, to make the game more realistic I can furtherly explore
* Add a health bar to account the health of the prisoners and to make the simulation more realistic.

1. **Future Work**

For future work, the model can follow more difficult by including different patterns of built-in walls to replicate indoor or outdoor designs of real-life buildings such as prisons, resorts, houses, apartments etc; in order to maximize security.

For example, “*According to the U.S. Bureau of Justice Statistics”* every year there is approximately 2,000 prison escapees. Which estimates at least 3% of prisoners behind the bars have the chance of escaping. The design of my simulation can be used to reduce the number of escapees each year by calculating the possible escape routes, prisoners are taking when the security team turn their back. It is simple to say to increase the number of staff to maximise security, however there is always a possibility of escaping

To do this I must simply fiddle with ‘Class method: Class Program for Box 1 and Box 2’ as the functionality of ‘Box.py’ has been made and imported to ‘FOP-Assignment.py’. To increase the number of restrictions, I can add more Boxes to increase the level of difficulty and adjust the size parameters length and width.

Thus today, this concept will be very utile for real-life applications if used correctly.

1. **References** 
   * A Newbie Guide to pygame — pygame v2.1.1 documentation. [2022 May 1] www.pygame.org. Available from: <https://www.pygame.org/docs/tut/newbieguide.html>
   * Sweigart A. Making games with Python & Pygame. S.L.: Al Sweigart, Cop; 2012.
   * Fonts 1001. Charybdis Font · 1001 Fonts .1001 Fonts. 2020. [2022 May 4]. Available from: <https://www.1001fonts.com/charybdis-font.html>
   * Police Dog.Pngplay.com. 2022. [2022 May 1]. Available from: <https://www.pngplay.com/wp-content/uploads/12/Clip-Art-Dog-Transparent-Free-PNG.png>
   * Police Goose.Freeiconspng.com. 2022 [ 2022 May 1]. Available from: <https://www.freeiconspng.com/uploads/goose-png-clipart-9.png>
   * Alarm.Vexels.com. ‌2022. [2022 May 1]. Available from: <https://images.vexels.com/media/users/3/149786/isolated/preview/c6bdfa7f8b99bb7e9c04b428cb670fec-firefighter-alarm-bell-illustration.png>
   * Pygame Zero Documentation Release 1.2.1 Daniel Pope. 2021. [2022 May 3]. Available from: <https://pygame-zero.readthedocs.io/_/downloads/en/1.2.1/pdf/>
   * Holzer R. Pygame tutorial Documentation. ‌2021. [2022 May 3]. Available from: <https://readthedocs.org/projects/pygame/downloads/pdf/latest/>
   * Wikipedia. 2019. *Qincheng Prison*.Contributors. Wikimedia Foundation; Available from: <https://en.wikipedia.org/wiki/Qincheng_Prison>
   * Mcgugan W. Beginning game development with Python and Pygame : from novice to professional. Berkeley, Ca: Apress ; New York, Ny; 2007
   * Quazi Nafiul Islam. Mastering pycharm. Packt Publishing Limited; 2015.
   * Curtin University. Practical Test 3. 2022.

**Required Softwares:**

* Inspiration 10 IE
* PYCharm IDE
* OneNote for Windows 10
* Google Chrome 🡪 Mydesktop 🡪 VMware Horizon 🡪 Python 3
* Word
* Excel