Università degli Studi di Pavia

Bachelor's in Artificial Intelligence

The Cannon Game

Final Exam Project

A.Y. 2023/24

Alessandro Di Stefano

Lorenzo Migani

Computer programming, Algorithms and Data Structures,

Module 1

Project Report

Goals of the project:

1) **Make the game easy to understand**:
   - Accessibility was a key consideration. The game was designed with a clear and intuitive user interface, accompanied by simple controls. We provided a help screen to guide new players through the mechanics. The goal was to ensure that players could quickly grasp the game's objectives and controls, making it approachable easily.
2) **Make the game playable at (almost) any resolution**:
   - To cater to a wide range of devices and screen sizes, the game was designed to be resolution-independent. This involved creating scalable UI elements and ensuring that gameplay elements remained consistent across different resolutions. The use of Kivy's layout management tools helped achieve this adaptability.
3) **Make the game aesthetically pleasing**:
   - Visual appeal was another critical aspect. We aimed to create a game that was not only fun to play but also visually engaging. This involved selecting a coherent art style. The aesthetic choices were made to complement the game's theme and enhance the overall player experience.

Design Choices:

The most creative design choice we made for this project is arguably the way levels are structured. Inspired by rogue-like games, we decided to give different layouts to levels 1 through 3: more precisely 8 types of level 1, 6 types of level 2 and level 3). Keeping the

4th level the same to make sure that if players fail to complete the game at their first try, they would be able to get to the same level again, making it easier over time.

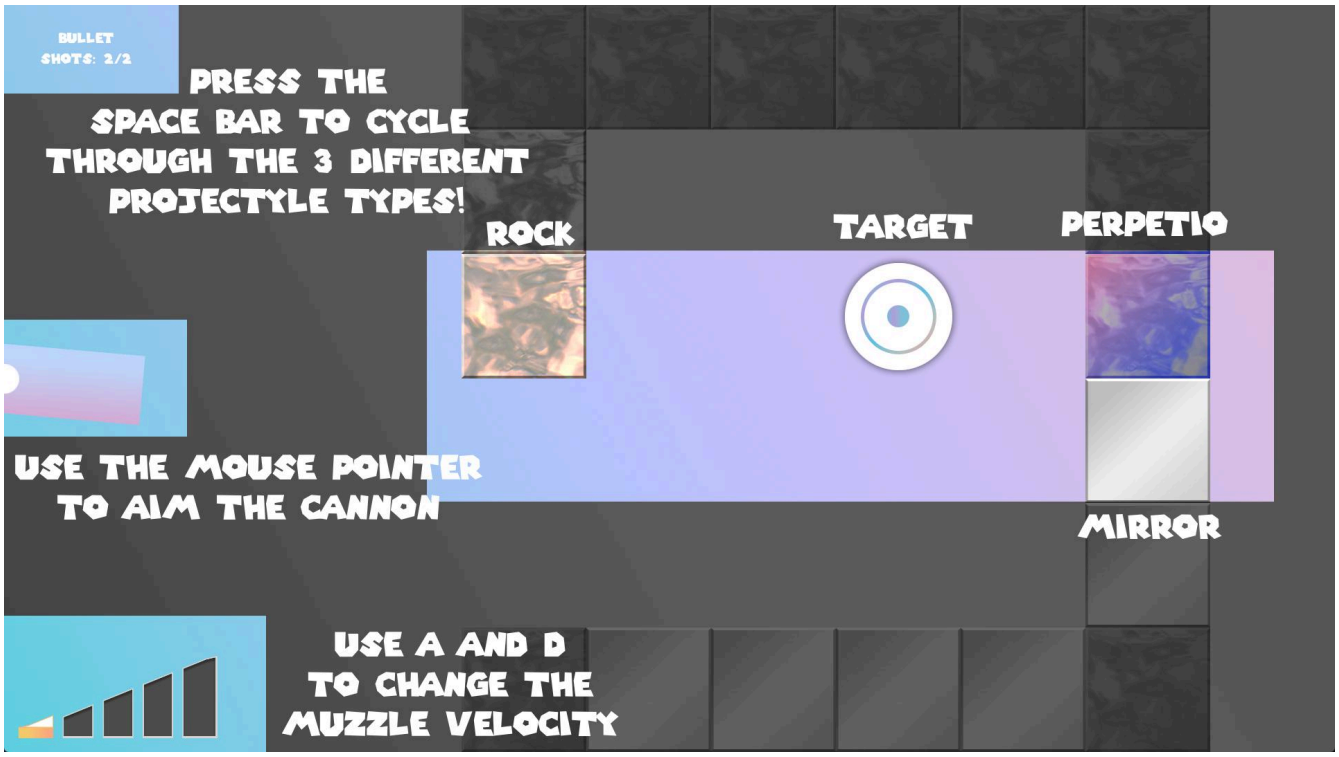
The design structure revolves around a modular and adaptive game architecture, where each level type is a self-contained module. This approach not only aids in maintaining the integrity and flexibility of the game but also facilitates future updates and expansions, such as adding new level types or game elements.

The level builder system randomly selects from the predefined layouts for levels 1 to 3 at the start of each level session. This randomness adds an element of surprise and replayability, encouraging players to adapt their strategies. The consistent layout for level 4 acts as a climactic challenge, providing a balance between unpredictability and learning curve.

Main screens in the game

When the game is opened you see the following screen and you have multiple choices,

you can click on HELP button where there are fundamental and necessary informations about the game and on how to play it,



Or you can click on the HALL OF FAME button to check the name of the 3 players that have the best score of all time.

Or you can click on the PLAY button, where another screen will be opened and you can choose to load an already started game or start a new game. If you choose to start a new game you will have to enter your name and then click on the “CLICK HERE TO START” button.

How levels are made

Our game is made of tiles, and every level is made of an 8 x 6 grid, to create a level you have to write a string of 48 char + a number that indicates the number of shots the player has in the level.

The string can have the following letters:

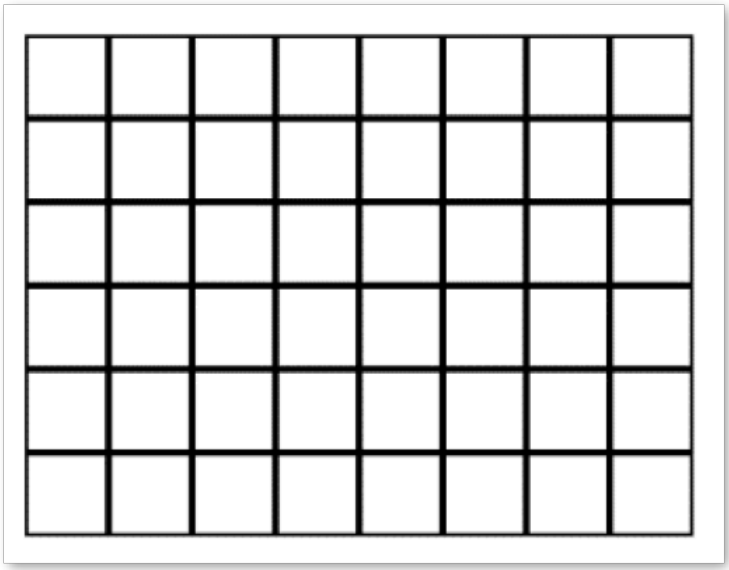
- ‘r’ that indicates the presence of a rock
- ‘t’ that indicates the presence of the target; in case there are more than one 1 in the string, all the ones that are after the first one are transformed into ‘n’
- ‘n’ is the char that indicates the absence of a particular tile
- ‘m’ indicates the presence of a mirror
- ‘p’ indicates the presence of perpetio

It is important to say that the space near the cannon cannot have any particular block and it is not considered in the grid.

For example to build the level

‘pnnnnnnnnpnnnnnnnnpnnnnnnnntnnnnnnnnpnnnnmmnnpnn3’ we start from the bottom left corner to the bottom right corner, then we go up by one tile starting from the left and going to the right and so on.

This is the grid we used to design levels:



Part of the code we used to create levels:

```
for i in range(48):
    x_pos = base_x_pos + ((i % 8) * tile_width)
    y_pos = base_y_pos + ((i // 8) * tile_height)

    element = leveldata[i]

    if element == 'n':
        continue

    if element == 't':
        if t_found:
            element = 'n'
        else:
            t_found = True

    if element in textures:
        level_canvas.add(Color( *args: 1, 1, 1, 1))  # Set the color
        texture = CoreImage(textures[element]).texture
        level_canvas.add(Rectangle(texture=texture, pos=(x_pos, y_pos), size=(tile_width, tile_height)))

        if element in 'rtmp':  # the collidables: rock, target, mirror, perpetio
            self.collidables.append({
                'type': element,
                'x': x_pos,
                'y': y_pos,
                'width': tile_width,
                'height': tile_height,
                'coord': i
            })
```

Characteristic of building blocks and target

Building Blocks

Each building block has a unique feature that makes it different from the other ones apart from the texture:

- The Rock referred to as the letter ‘r’ in the level design is the only block that can be destroyed and it can be both destroyed by the bullet and by the bomb, the difference between the two is that if a rock is hit by a bomb and has other blocks of rock around it they will be destroyed as well(within a radius of 2 tiles), instead if a rock is hit by a bullet it will be the only rock that is destroyed.
- The Mirror referred to as the letter ‘m’ in the level design is the only block where the laser can bounce. All other projectiles that hit the mirror will disappear. This building block is very important to make the game more interesting and fun to play.

- The Perpetio referred to as the letter ‘p’ in the level design is the only block that can never be destroyed no matter what projectile hits it, and in case of collision the projectile will disappear.

  Photos of the building blocks:

  from the left to the right: rock, mirror, perpetio

Target

Target is a fundamental element of the game. When hit by a projectile a visual cue will prompt the player to either keep playing or leave the game and get back to it another time.

Photo of the target:

Characteristic of projectiles

There are 3 different projectiles and each of them has been designed to have a different function in the game.

First thing first there is an important distinction to make between projectiles with or without a mass: bomb and bullet have mass and suffer gravity, laser doesn't have a mass and doesn't suffer gravity. To keeps things easy but effective we calculated the gravity that an object experiences by: mass *9.81. these are the other differences between the projectiles

- The bullet is considered the most versatile projectile in the game: it is able to destroy the rock it collides into and since it has a small texture and a small mass of 0.1, it doesn't suffer much from gravity and it doesn't collide against unwanted tiles, making it is useful in situations where you need a slight parabolic motion to hit the target or a rock.
- The Bomb is the most destructive projectile since it has an explosion radius of 2 and it is able to destroy multiple rocks at once. Differently from the bullet it has an important mass of 0.5, so it suffers more gravity and it also has the biggest texture in the projectile class making it more difficult to hit the wanted object.
- The laser is the only mass-less projectile in the game and it is the most difficult to use. To get the perfect bounce some training and trials are usually needed.

A further difference between the objects is that the velocity of the bullet and bombshell can be modified by changing the muzzle velocity in the bottom left of the screen using A to lower it and D to increase it, instead the laser has a constant velocity that can't be changed by the player.

Coding of the projectiles

Projectiles in this code are implemented as different classes based on the Projectile class.

Projectile Class: This is the base class for all projectiles, when a Projectile is created, it is initialized with a position (pos), an angle (angle), a velocity (velocity), and optionally a mass (mass). The initial velocities in the x and y directions (dx and dy) are calculated using trigonometry based on the angle and velocity.

- Movement: The move method updates the projectile's position based on its velocity and simulates the effect of gravity if the projectile has mass.
- Graphics: The update_graphics method is designed to be overridden in subclasses to update the visual representation of the projectile.
- Removal: The remove_projectile method is intended to be overridden to handle the removal of the projectile from the screen and relevant lists.

Projectile Movement

Each projectile's movement is updated periodically (every frame) by the update_projectyles method, which is scheduled to run at a set interval (30 times per second).

- Move Method: This method updates the projectile's position based on its velocities (dx, dy). If the projectile has mass, the effect of gravity is applied by reducing the vertical velocity (dy).
- Boundary Check: If the projectile moves off-screen, it is removed.

Collision Detection

Collision detection ensures that when a projectile intersects with other objects, appropriate actions are taken.

- Check Collisions: The check_collisions method iterates through all projectiles and checks for collisions with defined collidable objects.
- Is Colliding: This method checks if the bounding boxes of the projectile and a collidable object overlap.
- Handle Collision: Depending on the type of projectile and collidable object, different actions are taken (check ‘characteristic of bullets’ section).

Special Cases for Collisions

- Reflection of Lasers: The reflect method in the Laser class adjusts the laser's direction when it hits a mirror, based on the orientation of the mirror (vertical or horizontal).
- Explosion Radius: The bomb_explosion_collisions method checks if objects fall within the explosion radius of a Bombshell and handles their destruction.

Important Informations to know before starting to play + Main commands

There are some other informations that could be useful to fully understand the game:

- If we start a new game with a name ‘x’ and after completing the first level we save the game to finish it later, when we return to the game, finish the next level, and save again, the first save will be deleted because it will have a smaller score than the new save.
- If we start a new game with a name that has already been chosen before and we save the game, the already existing game will be deleted even though it has a higher score.
- When we start a level in the top left of the screen there will be the name of the projectile that is selected and under it there will be the number of remaining shots and the total shot for the game.
- To move the cannon we use the mouse, to shoot we left click with it.
- To change the selected projectile we use the spacebar.
- The formula to calculate the score is the following: Score = ((remaining shots×10)+10)×10×current level.

Tools and resources:

- **Programming Language**: Python 3.9

- **Framework**: Kivy 2.3.0

- **Physics**: Python math library for projectile calculations.

References:

- Kivy Course - Create Python Games and Mobile Apps

Community resources:
Stack Overflow and Reddit were invaluable for troubleshooting and exploring various implementation approaches. These platforms offered insights into best practices and creative solutions to common challenges.

Testing and Results

- **Usability Testing**: Gathering feedback on the user interface and controls to refine the player experience.
- **Gameplay Testing**: Iterating on game mechanics to balance difficulty and ensure a rewarding player progression.

The results were positive, with testers noting the game's responsiveness and ease of understanding. However, some areas were identified for potential improvement.

Possible improvements

Obviously, we could add all the optional obstacles.
In a different way, expanding the game with additional levels would provide more content for players. An infinite mode, focusing solely on achieving the highest score, could cater to competitive players looking for a challenge.