Computer Programming
Exam

SHADOW CANNON

PRESENTED BY

**CLAUDIA MARIA CARBONI AND
SUSANNA MAZZOCCHI**

Bachelor of Science in
Artificial Intelligence

Agenda

Short Introduction

- Unique gaming experience combining strategy, physics, and precision
- Player is the last Master of the Shadow Cannon
- Journey through four thematic realms: Hell, Abyss, Cursed Lands, Space
- Inspired by classic artillery games
- Control a cannon to hit targets while navigating obstacles

Goals

- Create an engaging and challenging game
- Progressive difficulty and complexity through four levels
- Variety of weapons (bomb, bullet, laser) and physical mechanics
- Encourage strategic thinking
- Complete the game with a limited number of shots and competitive time

Using:

- Programming Language: Python (version 3.12.0)
- Framework: Kivy (version 2.3.1)

Our Journey

Initial Prototype:

- Implemented essential functionalities
  - Cannon
  - Simple projectile
  - Target
- Identified potential problems and gained hands-on experience

Basic Mechanics:

- Developed physics system for projectile movement
- Implemented basic collisions and target detection

Level Expansion:

- Perfected the first level
- Incrementally added new mechanics and levels
  - Shotgun, laser, bats, mirrors, reflections

Optimization and Bug Fixing:

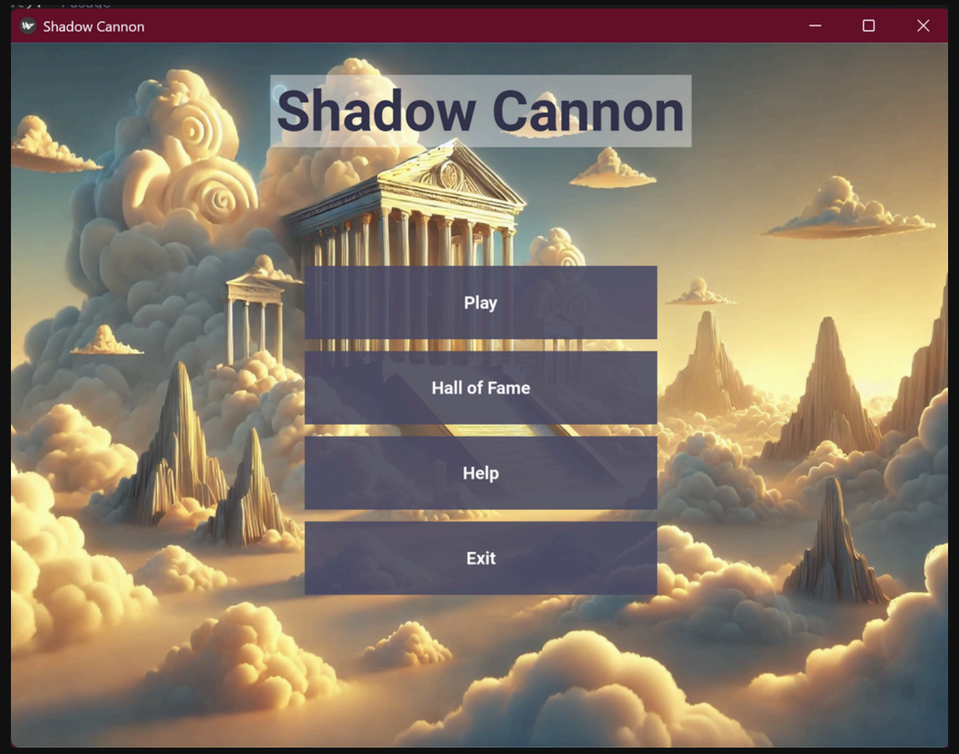
- Focused on optimization and balancing difficulty
- Fixed bugs to ensure smooth gameplay

Back to Agenda

Gameplay and Game Mechanics

Main Mechanics:

- **Shooting**: Player controls the cannon to fire different types of projectiles (bomb, bullet, laser).
- **Projectile Movement**: Projectiles follow specific physical behaviors (parabolic motion for bombs and bullets, linear for lasers).
- **Collisions:** Projectiles interact with obstacles (rocks, mirrors, bats) and targets, triggering specific responses (destruction, reflection).
- **Movement of Objects**: Targets like dragon and spaceship move according to game rules.



Back to Agenda

Gameplay and Game Mechanics

Separation of Logic and Interface:

- Python for game logic
- Kivy for graphical interface
- Clean and organized code

Controls:

- Use keyboard to adjust cannon angle and velocity
- Select projectile type (bomb, bullet, laser)

Scoring System:

- Points based on completion time
- Hall of Fame to track high scores

Back to Agenda

Additional Features:

Level Design:

- Four thematic levels with distinct visual themes and increasing complexity.
- Each level introduces new mechanics and challenges.

User Interface:

- Intuitive and consistent interface across levels.
- Popups for feedback and information (e.g., level completion, errors).

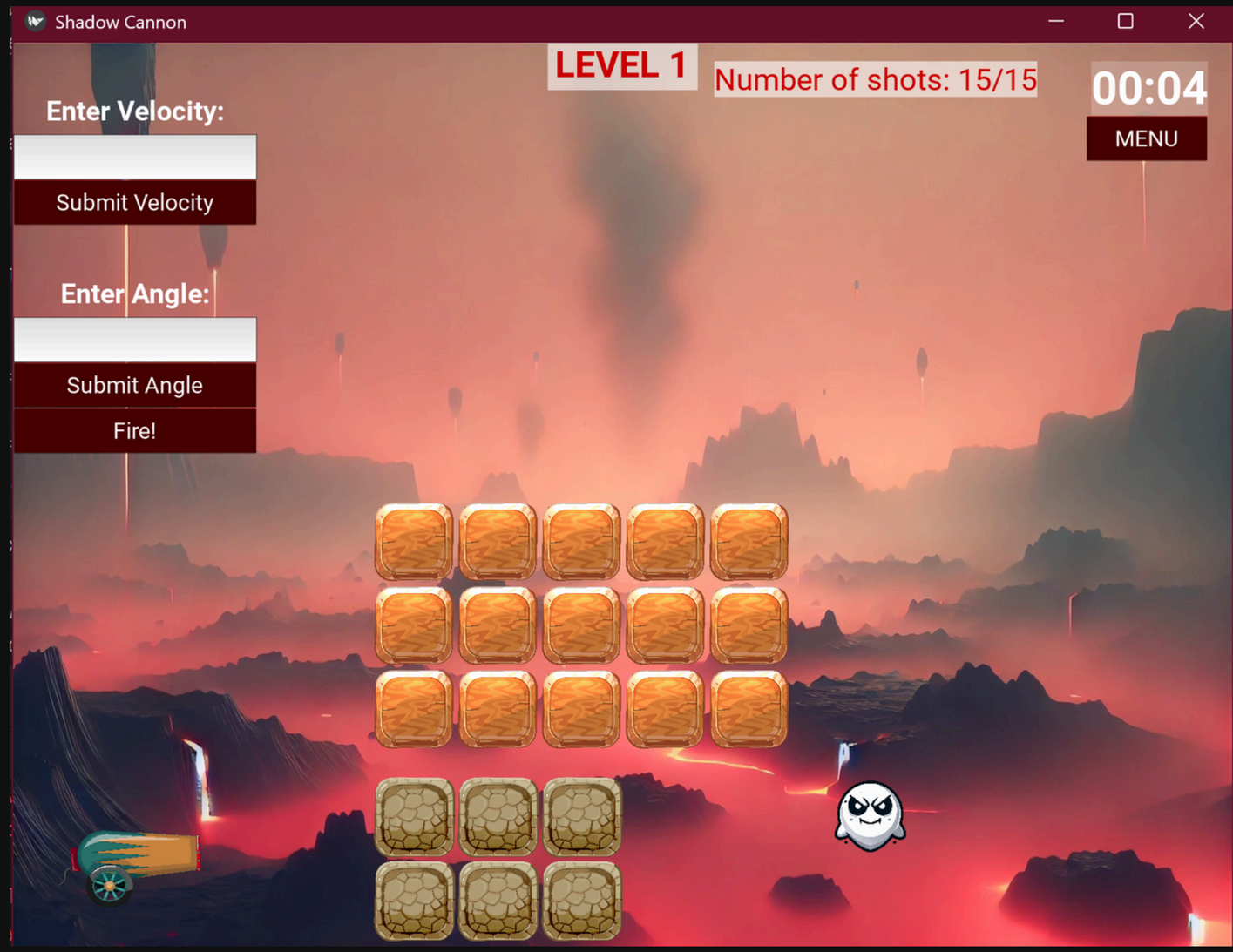
Audio and Visual:

- Background music to enhance immersion.
- Carefully chosen game images to enhance visual appeal.

Back to Agenda

Level 1: Hell

- **Projectiles**: Bombs.
- **Obstacles**: Destructible rocks, indestructible perpetuities.
- **Gameplay**: Use bombs to destroy rocks and hit the ghost target.
- **Difficulty**: Low (Introduction to basic mechanics and 15 shots available).



Back to Agenda

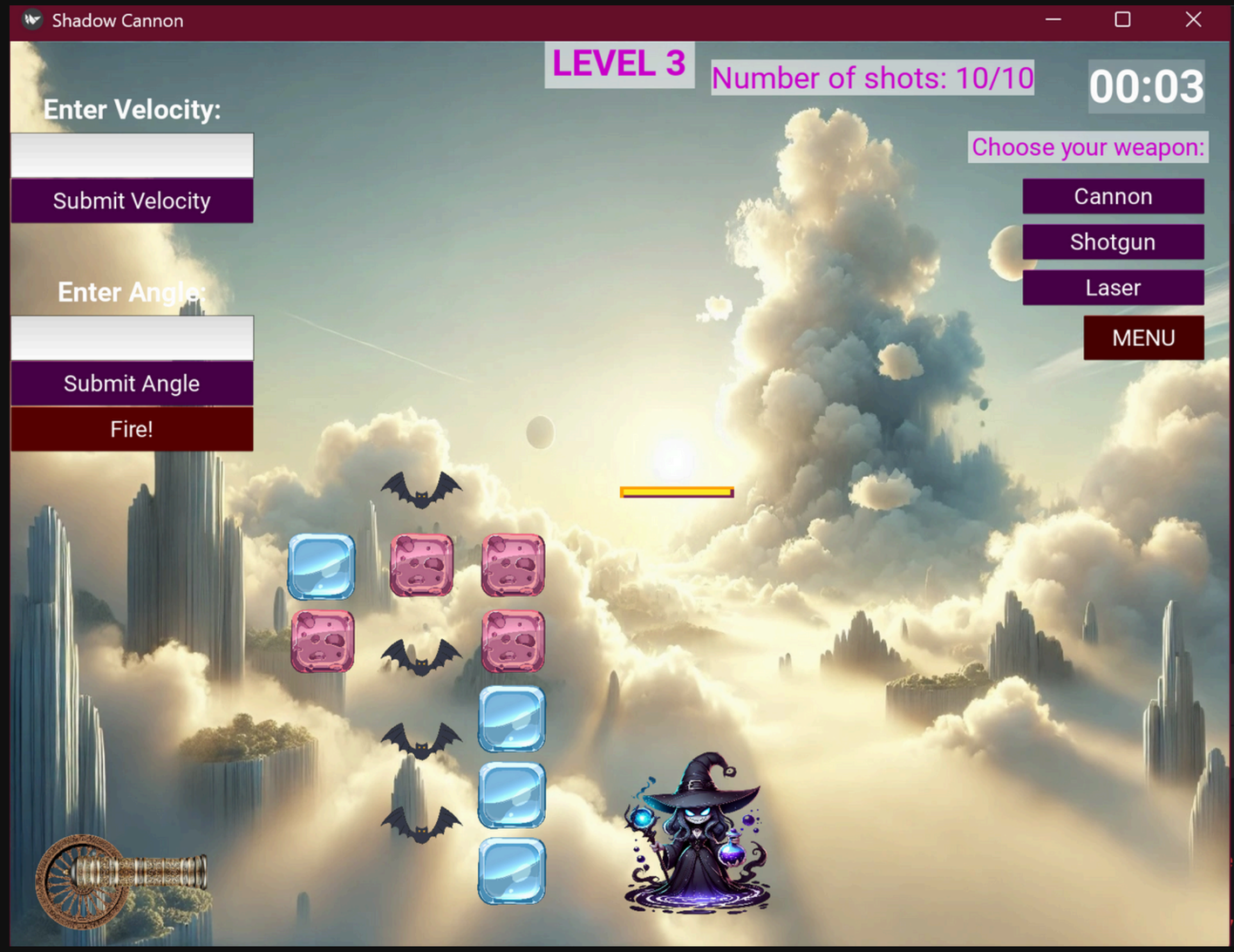
Level 2: Abyss

- **Projectiles**: Bombs, shotguns.
- **Obstacles**: Bats, rocks, indestructible perpetuities.
- **Gameplay**: Use shotguns to manage bats and bombs to clear rocks and hit the dragon target.
- **Strategy**: Choosing the correct angle and velocity is crucial.
- **User Interaction**: Switch between bomb and shotgun projectiles.
- **Difficulty**: Medium (Increased complexity with new obstacles and moving target).

Back to Agenda

Level 3: Curse

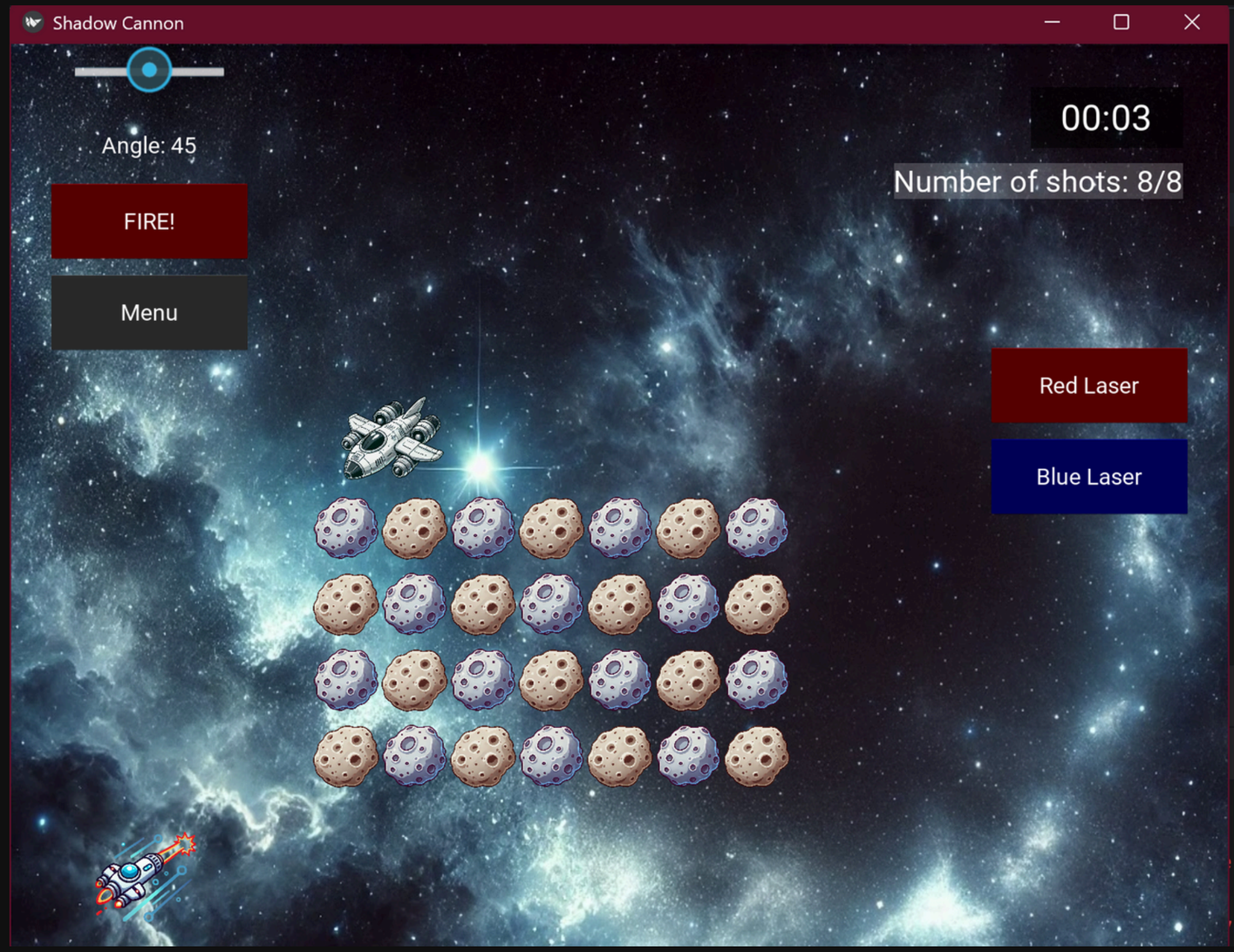
- **Projectiles**: Bombs, shotguns, lasers.
- **Obstacles**: Bats, rocks, indestructible perpetuities, mirrors.
- **Gameplay**: Use bombs to destroy rocks, shotguns to manage bats, and lasers to hit the witch target.
- **Strategy**: Choosing the correct angle and velocity is crucial.
- **Tip**: Plan your shots to use laser reflections off mirrors to hit the witch target.
- **Difficulty**: High (Advanced mechanics with laser reflections).



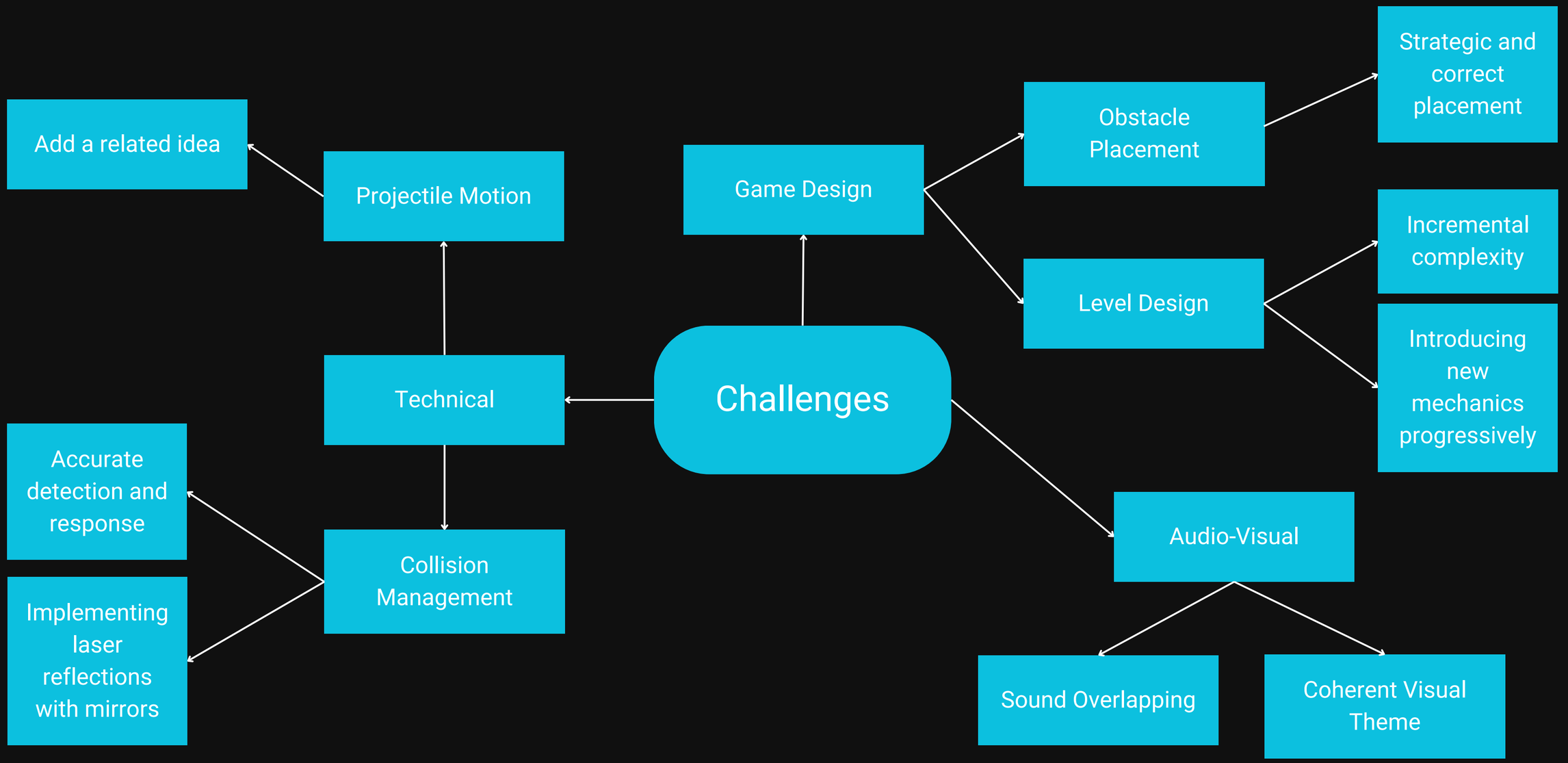
Back to Agenda

Level 4: Space

- **Projectiles**: Red lasers, blue lasers.
- **Obstacles**: Asteroids, spaceship.
- **Gameplay**: Use red lasers for the spaceship and blue lasers for the asteroids to clear your path.
- **Difficulty**: Very High (Maximum complexity with dual lasers, very limited numer of shots and and moving target).



Back to Agenda


Challenges
Technical
Projectile Motion
Add a related idea
Collision Management
Accurate detection and response
Implementing laser reflections with mirrors
Game Design
Obstacle Placement
Strategic and correct placement
Level Design
Incremental complexity
Introducing new mechanics progressively
Audio-Visual
Sound Overlapping
Coherent Visual Theme

Our Approach

Incremental Development:

- Adding features one at a time.
- Building upon previous levels.
- Focused on optimizing performance and balancing difficulty.

Extensive Debugging:

- Used numerous print statements to identify and resolve issues.
- Conducted various tests by changing parameters and functions to find optimal solutions.

Continuous Improvement:

- Regularly refined and enhanced game elements based on testing and feedback.
- Ensured consistent code quality through regular meetings and idea exchanges.

Back to Agenda

Code Structure

Main Components:

- **Cannon Levels**: Separate classes for each level (CannonLevel1, CannonLevel2, CannonLevel3, CannonLevel4).
- **Screens**: Classes for different screens (MainMenu, LevelSelection, Help, IntroStory, HallOfFame, CongratsScreen).
- **Game Logic**: Methods for shooting, collisions, and level resets.
- **UI Element**s: Pop-ups, labels, and buttons for user interaction.

File Structure:

- **main.py**: Main application file.
- **.kv Files**: Separate KV files for UI definitions (menu.kv, Level1Game.kv, etc.).
- **Images and Sounds**: Organized in folders for easy access.

Back to Agenda

Example Codes

```
def drop(self, dt):
    """
    Update the cannonball's position and handle collisions.
    """
    # Update cannonball position
    new_x = self.cannonball.pos[0] + self.velocity_x
    new_y = self.cannonball.pos[1] + self.velocity_y
    self.cannonball.pos = (new_x, new_y)

    # Apply gravity
    self.velocity_y -= 0.98

    # Check for collisions
    self.check_collisions()
```

```
def check_collisions(self):
    """
    Check for collisions with the ghost target, obstacles, and perpetual obstacles.
    """
    ball_left = self.cannonball.pos[0]
    ball_right = ball_left + self.cannonball.size[0]
    ball_bottom = self.cannonball.pos[1]
    ball_top = ball_bottom + self.cannonball.size[1]

    # Collision with the ghost target
    ghost_left = self.ghost.pos[0]
    ghost_right = ghost_left + self.ghost.size[0]
    ghost_bottom = self.ghost.pos[1]
    ghost_top = ghost_bottom + self.ghost.size[1]

    if (ball_right > ghost_left and ball_left < ghost_right and
            ball_top > ghost_bottom and ball_bottom < ghost_top):
        self.hit_target()
        self.reset_cannonball()
```

Back to Agenda

Our Creativity

- Ability to launch **three projectiles at once.**
- **Four Levels with Increasing Difficulty** to keep the game engaging and challenging.
- **Visual Design**: Easy-to-use, intuitive, and consistent design across all levels.
- **Thematic Consistency**: Recurring themes that enhance the storytelling and immersion.
- **Dual Laser Types in Level 4:** Red laser for asteroids and blue laser for the spaceship target, adding a layer of strategy.

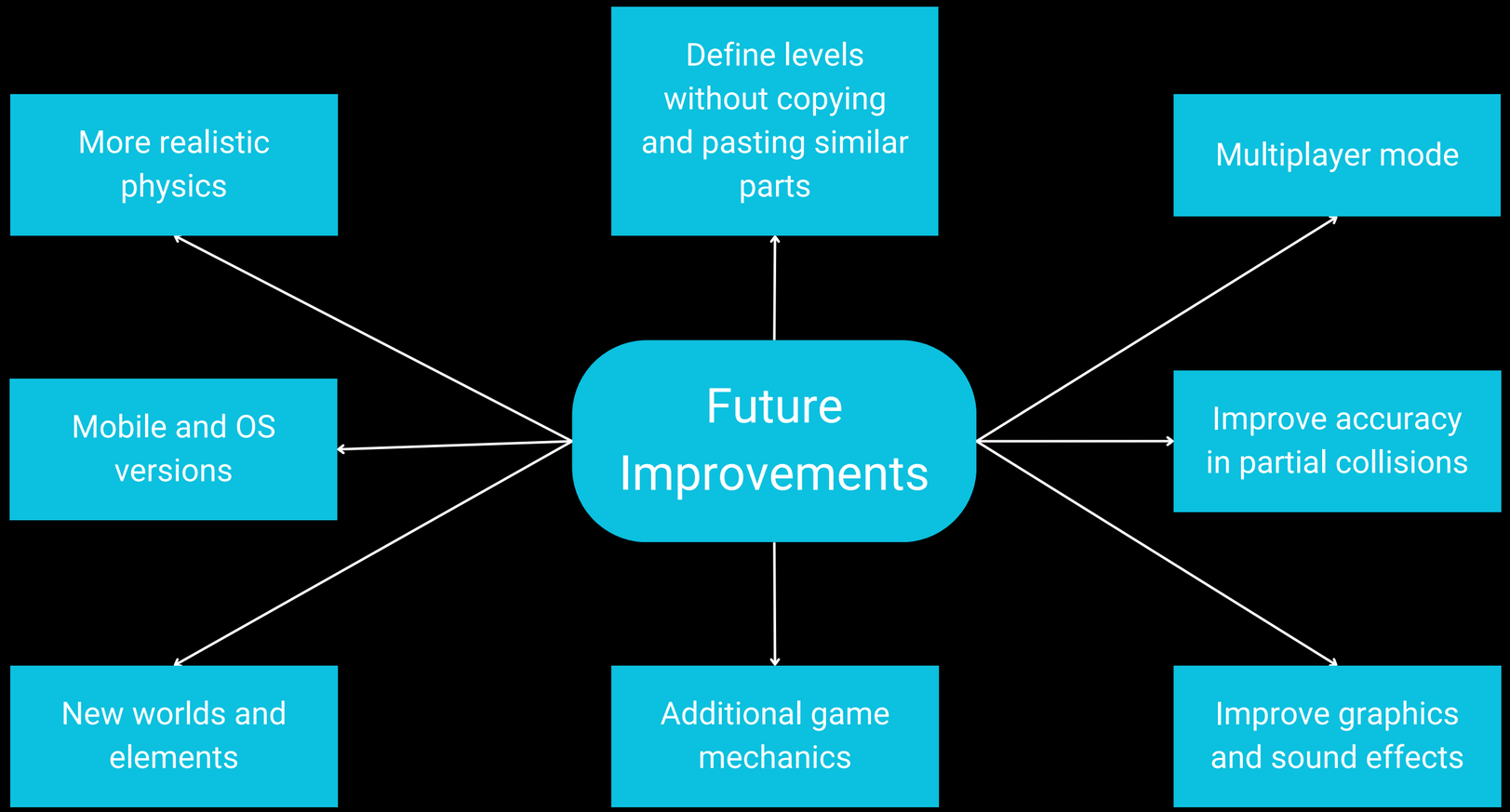
Our Creativity

User Experience:

- Stimulating **Background Music** generated using AI prompts to enhance immersion.
- **Pop-ups**: Informative pop-ups for various in-game events:
  - Hitting indestructible rocks.
  - Using the wrong projectile for obstacles.
  - Level completion prompts.

Additional Features:

- **Time Tracking**: Allows users to see their completion time, encouraging faster playthroughs.
- **Target-Specific Projectiles**: Different projectiles are effective against specific targets.
- The game is intuitive and user-friendly, with clear feedback and guidance.


Future Improvements
More realistic physics
Define levels without copying and pasting similar parts
Multiplayer mode
Mobile and OS versions
Improve accuracy in partial collisions
New worlds and elements
Additional game mechanics
Improve graphics and sound effects

Thank You!

BY:

CLAUDIA MARIA CARBONI (535421)
SUSANNA MAZZOCCHI (535996)

COMPUTER PROGRAMMING
BCs in Artificial Intelligence