

Overview

JUDDO-THE FIGHT is an interactive, action-oriented 2D shooter game built using Python and OpenGL (PyOpenGL). The project merges classic arcade gameplay mechanics with modern programming techniques, placing two stickman warriors against each other (or against the computer) in a dynamic, obstacle-filled arena. Players select characters, utilise special abilities, dodge obstacles, collect coins, and aim to outmanoeuvre and outgun their opponent across multiple rounds.

1. Technical Foundation & Libraries

The project leverages several core Python libraries:

- **math, random, time**: For calculations, random number generation (spawning, AI choices), and time-based actions (intervals, cooldowns).
- **OpenGL.GL, OpenGL.GLU, OpenGL.GLUT**: These modules provide the low-level graphics rendering pipeline for drawing shapes, handling input, and managing onscreen display. GLUT is particularly used for windowing and event loops.

2. Game Architecture

2.1 Window & Display

- The main game window is sized for landscape play (`1200x800` pixels), with double buffering and RGB colour mode for smooth rendering.
- Responsive design is enabled via `reshape`, ensuring the game scales with window size changes.

2.2 Game States

The game employs a **finite state machine** style for its flow control, with these key states:

- **Main Menu**: Entry point; lets the user choose Single Player, Multiplayer, or Exit.
- **Character Select**: Each player (or AI) chooses a character; disables already "dead" (used/eliminated) characters in later rounds.
- **Gameplay (Multiplayer/Singleplayer)**: The core battle arena phase.
- **Game Over / Final**: End of round/game, displays scores, winner, and options to restart or exit.

2.3 UI System

Buttons are completely custom-drawn with OpenGL primitives, supporting hover effects, enable/disable states, and contextual coloring. Each menu has its own button configuration, all with normalised (ratio-based) coordinates for easy scaling.

3. Core Gameplay Elements

3.1 Characters

Players can select from three distinct characters, each with unique attributes:

Character	Gun Type	Health	Damage	Speed	Firerate	Special Ability	Color
Reety	Uzi	100	19	2.0	0.25s	Rage (Rapid Fire)	Violet
Argha	Scar	100	20	2.5	0.5s	Gun-jam	Blue
Avishek	Sniper	125	50	1.75	0.75s	Laser	Cyan

Specials add tactical depth; for example, Rage halves the firerate for burst shooting, and Shield blocks damage for a short period.

3.2 Player Classes

- **Player Class**: Encapsulates position, movement, shooting logic, health, abilities, and AI (if applicable).
- **AI Player**: In Singleplayer mode, Player 2 is controlled by a basic AI that tracks, avoids obstacles, and shoots at intervals.

3.3 Movement & Controls

- Controls are mapped for both WASD (Player 1) and IJKL (Player 2), with additional keys for aiming adjustment and shooting.
- Movement is restricted by obstacles and window boundaries, using AABB collision logic.
- Aim can be adjusted up/down, and the length of the aiming line is modifiable for precision shots.

3.4 Shooting Mechanism

- Shooting is directional, based on the player's aiming angle.
- Projectiles travel linearly and are destroyed upon hitting a player, obstacle, or leaving the screen.
- Each projectile carries the shooter's damage value.

3.5 Obstacles

Obstacles are rectangular and strategically placed. They block player movement and destroy projectiles. Their positions are randomized (but hard-coded in this version), and collision checks use bounding box logic.

3.6 Coins

- Coins spawn at random positions, not overlapping obstacles, up to a maximum of 10 at a time.
- Collecting a coin slightly restores player health (capped to their maximum).
- Coins are visually rendered as yellow balls.

4. Rendering & Graphics

4.1 Drawing Primitives

- **Stickmen**: Rendered with lines (body, arms, legs) and circles (head), colored per character.
- **Weapons (Uzi)**: Custom polygon shapes and rectangles, with glowing barrel effects.
- **Thruster Effects**: Animated jet trails at the feet, using transparency gradients.
- **Projectiles**: Small white balls.
- **Obstacles/Coins**: Rectangles and circles with distinct colours.

4.2 UI & Feedback

- Buttons feature hover and active feedback.
- Scores, health, round info, and coin counts are displayed using OpenGL bitmap fonts.
- Game Over and Final screens feature colored backgrounds and celebratory text.

5. Game Flow & Logic

5.1 Rounds & Elimination

- The game is played in up to 3 rounds.
- When a player is eliminated (health reaches zero), their character is marked "dead" and cannot be used in subsequent rounds.
- After each round, states reset (except for eliminated characters), and selection menus appear again.

5.2 Winner Determination

- After the final round, scores are tallied and a winner (or tie) is displayed.
- Players can return to the main menu or exit.

6. Input Management

- **Keyboard**: Movement, aiming, and shooting for both players.
- **Mouse**: Used for button clicks and, in some modes, to shoot (Player 2).
- **Hover Logic**: Mouse movement updates button hover states for UI responsiveness.

7. Code Architecture

- **Modular Functions**: The code is split into logical sections: drawing utilities, class definitions, UI components, collision checks, game state updates, and input/event handling.
- **Main Loop**: Managed via GLUT callbacks for display, reshape, keyboard/mouse events, and a timer updating the game state at ~60 FPS.

8. Notable Features & Enhancements

- **Visual Effects**: Transparency, animated thrusters, glowing gun barrels.
- **AI Opponent**: Rudimentary, but demonstrates obstacle avoidance and targeted shooting.
- **Health Restoration**: Coins introduce a risk/reward mechanic.
- **Preventing Duplicate Character Selection**: Dead characters are greyed out and disabled.
- **Responsive UI**: All buttons and UI elements scale with window size.

9. Extensibility & Suggestions

- **Future Improvements**:
- More advanced AI (pathfinding, prediction).
- Powerups, varied obstacles, and environmental hazards.
- Expanded character roster and weapon types.
- Sound effects and music.
- Save/load high scores or progression.
- Online multiplayer or networked play.

10. How to Run

- Ensure you have Python and the required libraries (PyOpenGL) installed.
- Run the script (`JUDDO-THE FIGHT.py`) from the terminal or an IDE.
- The game window will appear; use the keyboard and mouse as described above.

11. Summary

JUDDO-THE FIGHT is a vibrant, feature-rich, and well-structured Python game project that demonstrates a solid grasp of graphics programming, event-driven architecture, and game design fundamentals. It is both a playable game and a strong template for further experimentation and expansion, balancing technical sophistication with approachable gameplay.
