“🛸Galactic Interceptor 3D: Elite Edition”

Shafin Mahamud -23101109

"The last line of defense between order and total destructioin."

In *Galactic Interceptor 3D*, you step into the cockpit of the RX-78 "Vanguard," the most advanced spacecraft ever built. The galaxy is facing a cataclysmic event known as the "Starfall"—rogue geometric entities and fallen stars are collapsing into the civilized sectors, threatening to erase existence.

The Purpose: You are the Guardian. Your mission is not just to shoot, but to survive and endure. Every level cleared represents a sector saved; every enemy neutralized is a life preserved.

The Mental Outcome: This game is designed to induce a state of **"Flow."** The rhythmic destruction of enemies, combined with the neon-soothing visuals, provides a sense of control and focus. It transforms anxiety into action. When you win, the "Happy Face" reinforces a sense of accomplishment and dopamine release. When you lose, the "Sad Face" evokes empathy and the drive to try again, building resilience against failure.

2. Technical Feature Breakdown

Below are the **10 Core Features** of the game and the specific Python/OpenGL functions used to build them.

Feature 1: High-Fidelity Modern Spacecraft

A complex, multi-part 3D model representing the player, complete with engine thrusters, swept-back wings, and a metallic finish.

- **Function Used:** `draw_modern_ship()`
- **Implementation:** Utilizes `gluCylinder` for the fuselage, `glutSolidCube` for the engine block, and `glScalef`/`glRotatef` to assemble primitive shapes into a cohesive vehicle.

Feature 2: Dual Camera System (Tactical & Cockpit Views)

Allows the player to toggle between a cinematic Third-Person view and an immersive First-Person cockpit view.

- **Function Used:** `display()` (Camera Setup Section)
- **Implementation:** Uses conditional logic to switch parameters inside `gluLookAt()`.

    - *3rd Person:* Calculates offsets using `math.sin` and `math.cos` based on `cam_dist`.
    - *1st Person:* Places the camera directly at `player_x`, `player_z`.

Feature 3: Adaptive Difficulty & Variable Firepower

The game evolves with the player. In Level 1, the ship fires a massive 5-bullet spread to help beginners. As levels rise, weapon spread decreases and enemies become faster, demanding higher skill.

- **Function Used:** `fire_logic()` and `update()`
- **Implementation:**
    - *Firepower:* `bullet_count = max(1, 6 - level)` dynamically adjusts shot count.
    - *Enemies:* `max_enemies = 5 + (level - 1) * 2` scales the threat level.

Feature 4: "Ultimate" Cheat Mode

A hidden developer mode that activates a protective force field, auto-aiming projectiles, and a 360-degree "Spin Attack."

- **Function Used:** `draw_modern_ship()` (Shield rendering) and `fire_logic()` (Homing math)
- **Implementation:**
    - *Shield:* Uses `glutWireSphere` with `glEnable(GL_BLEND)` for a transparent energy effect.
    - *Homing:* Calculates vector magnitude `math.sqrt(dx*dx + dz*dz)` to normalize bullet velocity toward the nearest enemy.

Feature 5: Zoom & Camera Control

Players can dynamically zoom in or out in Third-Person view to adjust their field of view.

- **Function Used:** `mouse()` and `keyboard()`
- **Implementation:** Detects scroll wheel inputs (`btn == 3` or `4`) or bracket keys (`[ / ]`) to increment/decrement the global `cam_dist` variable.

Feature 6: Algorithmic Facial Expressions (Win/Loss)

Instead of using images, the game draws "Happy" or "Sad" faces pixel-by-pixel using the Midpoint Circle Algorithm to convey emotion at the end of a level.

- **Function Used:** `midpoint_circle_arc(r, cx, cy, mood)` and `draw_face()`

- **Implementation:** A manual implementation of the **Midpoint Circle Algorithm**. It selectively renders specific octants of the circle (e.g., `py < cy` for a smile) based on the `mood` parameter.

Feature 7: Heads-Up Display (HUD) & Interactive UI

A 2D overlay that displays real-time stats (Lives, Score, Level) and clickable interactive buttons for game flow.

- **Function Used:** `draw_hud_and_menus()` and `draw_button()`
- **Implementation:** Switches to 2D Orthographic projection (`gluOrtho2D`) temporarily to draw text and quads over the 3D scene, then switches back to 3D perspective.

Feature 8: "Never-Miss" Spin Mechanics

In Cheat Mode, the ship physically spins, and bullets are fired in a spiral pattern to clear the screen.

- **Function Used:** `update()` (Animation)
- **Implementation:** Updates `player_spin` variable every frame modulo 360. This variable is applied via `glRotatef` in the drawing phase.

Feature 9: Robust Game State Management

Handles the transitions between Playing, Level Completion, and Game Over states without restarting the application.

- **Function Used:** `advance_level()` and `restart_game()`
- **Implementation:** Resets entity lists (`enemies = []`, `bullets = []`) and updates state flags (`game_state = 1` or `2`). Ensures the difficulty curve resets on death but persists on level advance.

Feature 10: Clean Exit System

Allows the user to safely terminate the program from within the game UI.

- **Function Used:** `mouse()`
- **Implementation:** Detects clicks within the bounding box of the "EXIT" button and calls `sys.exit(0)` to close the OpenGL window and Python process gracefully.

More to be continued..