

Submission 2 Documentation

CanYouDrone is a drone flying simulator game where you need to fly the drone through checkpoints (Contact with the ground is allowed for now). The game is won when all the checkpoints have been visited in order. (Three checkpoints is already difficult enough)

(joystick is 0 indexed in software, but physically has numbers starting at 1)

Controls:

Thrust upwards:

Spacebar | Joystick Axis 3 | Gamepad Right Trigger

Tilt left and right (Roll):

A and D | Joystick Axis 0 | Gamepad Left X

Tilt forwards or backwards (Pitch):

W and S | Joystick Axis 1 | Gamepad Left Y

Turn on the spot (Yaw):

Q and E | Joystick Axis 2 | Gamepad Right X

Auto Level (VERY USEFUL, active by default)

B | Joystick Button 8 (7) | Gamepad Button A

Reset the drone to an upright position (resets checkpoints):

V | Joystick Button 3 (2) | Gamepad Right Bumper

Reset drone to start point (resets checkpoints)

R | Joystick Button 2 (1) | Gamepad Start

Switch between camera modes:

C | Joystick Trigger (0) | Gamepad X

Rotate Camera:

RMB + Mouse | Joystick Hat (12-15) | Gamepad Dpad

Camera Zoom:

Mouse Wheel

Reset Camera:

M | Joystick Button 6 (5) | Gamepad Right Thumb

Toggles Controls HUD Display:

H

Toggle ImGui HUD - DEV/DEBUG MODE (change inputs, access dev inspector)

F10 | Joystick Button 10 (9) | Gamepad B

Toggle drone telemetry (imgui) HUD (only if F10 HUD is active)

F8

Toggle Collision Wireframe

F3 | Joystick Button 7 (6) | Gamepad Left Bumper

Quit Game:

Escape | Joystick Button 9 (8) | Gamepad Button Back

We suggest using a controller, because flying with the keyboard is difficult. (Or even better, a joystick if you have one)

We have an entire rebinding system written, so if you have any controller or joystick and the default mapping is weird then feel free to rebind it. I didn't get around to rendering this myself, so that is still (and might stay) in ImGui.

F10 (top edit bar shows up, top left "Tools" -> "Input Editor")

Maps/Actions are categories,

Let's say you have a joystick where the throttle goes from 0 to 1 so our default binding is wrong for you.

(my joystick goes from 1(no power) to -1(full power) on axis 3, so i need to * -0.5 and offset by 0.5 to get to a 0-1 value)

Then you go to the left menu "Drone" -> "Throttle" – center menu -> "Joystick Axis 3" – right menu -> "Node Processors" and edit this binding there. e.g. remove the scale and offset.

This system should allow any strange combination of input devices you can throw at it.

Implemented Effects and Features:

refer to "[Implementation.md](#)" inside the project root for more detail

- 3D geometry and textured models are implemented through glTF model loading. The drone is rendered as a textured 3D object. The map and rotor blades only have solid colors.
- Spinning rotors are implemented as visible animated moving parts on the drone model.
- Directional and point lighting are implemented. The world is illuminated by a directional light, while the drone also has 1 point light with shadows and 4 point lights without shadows (Specular Map)
- Shadow mapping with PCF and Omnidirectional is working. The map and drone cast and receive shadows.
- Physics with Bullet are implemented. The drone uses rigid body simulation and reacts to forces, torque, gravity, and collisions.
- Adjustable parameters are implemented for testing and balancing. Physics values such as mass, rotor force, rotor position, can be changed during runtime. (F10 Menu) But there is also a config file to adjust window settings.
- Input abstraction is implemented and allows for intuitive controls.
- We implemented HUD functionality to overlay Text in the game (Used for FPS counter and key bindings)
- For now, telemetry and input rebinding is implemented through a DearImGui overlay (F10 and F8) to make debugging easier. There is no need to interact with it as basic key bindings are already set
- Advanced Gameplay?
- LUTs

There is a Windows and a Linux (tested on Fedora) port for the game.

Depending on the hardware, starting the game might take a few seconds, because we calculate the collision mesh from scratch during startup based on the map.

Libraries Used:

- GLFW for window creation and input handling

- OpenGL for rendering
- GLM for vector, matrix, and quaternion math
- Bullet Physics for rigid body simulation and collision detection (<https://github.com/bulletphysics/bullet3>)
- Dear ImGui for debug and telemetry (<https://github.com/ocornut/imgui>)
- Tracy Profiler to look at our performance problems (<https://github.com/wolfpld/tracy>)
- Model loader (<https://github.com/jkuhlmann/cgltf>)
- TrueType Font loader for in game text (<https://nothings.org/stb/>)
- Image loader for the loading screen and LUTs (<https://nothings.org/stb/>)
- "Desert Race Game Prototype Map V2" (<https://skfb.ly/6ZrEy>) by Batuhan13 is licensed under Creative Commons Attribution (<http://creativecommons.org/licenses/by/4.0/>).
- Drone made by me (Marco Zechner)