Computer Graphics Coursework

InArcadia

Tyler Graves u2202289
Year 3 Mathematics

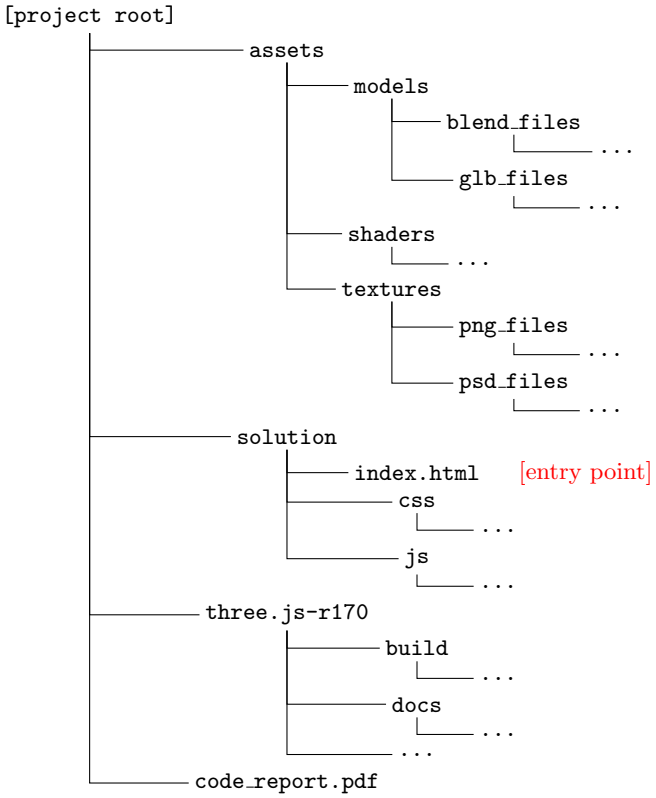
January 17, 2026

Contents

1 Instructions

1.1 Project Structure

The .zip file submitted has the following structure (as specified in the coursework brief) :



`./solution/` contains .js, .html, and .css files.
`./assets/` contains .blend, .glb, .glsl, .png, and .psd files.
`./three.js-r170` contains version r170 of three.js.
This structure must be preserved in order for the program to run without error.

1.2 Launching the Game

After unzipping the submitted .zip file, open a terminal in the project root directory (containing `./assets, ./solution, ...`). Here, run a local server - assuming Python 3.x is installed, this can be done by simply running:

```
python -m http.server 8000
```

If port 8000 is already in use, choose an appropriate alternative. The following message should be displayed:

```
Serving HTTP on :: port 8000 (http://[::]:8000/) ...
```

Now, open a browser and, in the address bar, type `http://localhost:8000/solution/`. You should see the game's main menu screen.[1]

[1] This was tested on the DCS Linux machines with Google Chrome on 16.01.26

2 Project Breakdown

2.1 Game Overview

In the game, you play as a video game character inside of an arcade machine. You spawn out of a mysterious door in the sky and must reach the end of the track to complete the level. You navigate the environment by walking, jumping, sprinting, and dashing, and can gain more information about your surroundings by toggling between two different camera views: a first person view from the character's perspective, and a third person view from the arcade machine screen.

This gameplay was inspired by the movement in games like Ghost Runner, and the concept of being stuck in an arcade machine was inspired by the brief requesting two camera angles.website:patronus

2.2 High-Level Code Structure

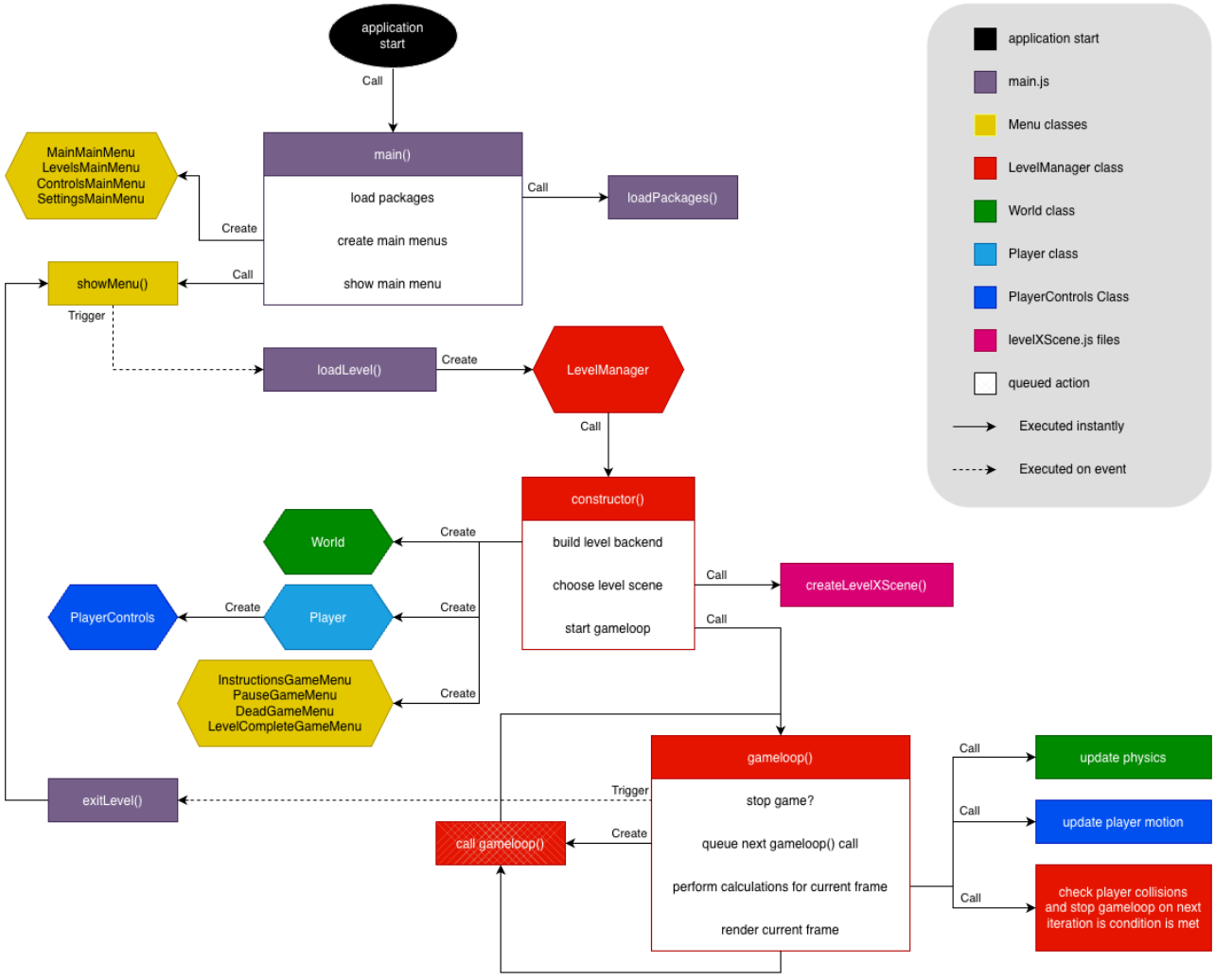


Figure 1: Simplified game lifecycle flowchart

Figure 1 shows a simplified version of the game's lifecycle - from startup, to loading a level, to exiting back to the main menu.

When the application is started, main() is called in main.js which loads assets, creates and shows the main menu, and awaits a response. Once the user selects a level, the menu triggers the loadLevel() function in main.js which creates a LevelManager.

The LevelManager then calls its constructor() function which initialises the level by creating World, Player, and in-game Menu objects and calling the appropriate createLevelXScene() function. It then starts the gameloop.

The gameloop first checks if the game should stop. If so, it triggers exitLevel() in main.js and shows the main menu. Otherwise, it queues the next gameloop() function call, and performs calculations for the current frame. The current frame is then rendered, and the queued gameloop() function call is executed, creating a loop.

2.3 Detailed Code Breakdown

2.3.1 Level Generation

TALKING POINTS FOR EACH FILE: - buildWorld - makes two cameras and adds to array - creates a scene with fog - physics initialisation - level generation and object spawner in one - uses classes from object spawner to create all scene objects in the level - this includes: screen, track, spawn platform, outofboundsplatform, levelcompleteplatform, imageplates - level manager - created and takes over from main when level is selected - creates gameworld, player, level environment - handles level events such as game pausing, player dying, camera switching, player (re)spawning - controls the gameloop - menus - handles the main and game menus - creates classes that are used in main and level manager to create levels - all the classes do is toggle html screens on and off and provide logic for the buttons - also contains the HUD logic - player - creates the player's physics body, debug mesh, and player character - handles (re)spawning of character - creates player controls which handles all player movement - defines listeners for mouse and key input and contains movement logic, including sprint and dash mechanics - updates player location and camera locations based on input

2.3.2 html, css, and glsl

- very briefly cover html and css. go into detail a bit more on glsl (mention why it is in assets)

2.3.3 textures and models

- briefly explain pipeline for making textures: goodnotes -¿ photopea. Explain blender model

Images/P1040028.JPG	Images/P1040568.JPG	Images/P1050447.JPG	Images/P1040713.JPG
Images/P1050382.JPG	Images/P1050184.JPG	Images/P1050129.JPG	Images/P1050086.JPG
Images/P1040256.JPG	Images/P1040948.JPG	Images/P1040941.JPG	Images/P1040933.JPG

Images/P1040924.JPG	Images/P1040923.JPG	Images/P1040914.JPG	Images/P1040894.JPG
Images/P1040873.JPG	Images/P1040739.JPG	Images/P1050410.JPG	Images/P1040705.JPG

Images/P1050477.JPG	Images/P1040552.JPG	Images/P1040358.JPG

Images/P1040264.JPG	Images/P1040984.JPG	Images/P1040167.JPG	Images/P1040159.JPG
Images/P1040139.JPG	Images/P1040135.JPG	Images/P1040131.JPG	Images/P1040116.JPG
Images/P1050549.JPG	Images/P1040026.JPG	Images/P10203042.JPG	Images/P1020281.JPG
Images/P1020276.JPG	Images/IMG_9214.jpeg	Images/IMG_4954.JPG	