

# Lego Racers: Remake++

---

An attempt to remake my childhood favourite Lego Racers, with added features, including intelligent enemies, powerups and obstacles.

## Competitive Analysis

The first racing game to analyse will of course be **Lego Racers** itself. This will naturally be a hard game to beat in terms of playability, mechanics and features. However, this remake++ version will not only incorporate the good features of Lego Racers, but also add on some features:

Similarities	Differences
Powerups	Intelligent enemies and obstacles
-	Different camera views
Tracks, racecar & character selection	Will not be able to customise fully
<i>Customisable racecars (KIV)</i>	
<i>Enemy cars (KIV)</i>	Single player mode
Supports Windows	Coded in Python, so <b>multiplatform!</b>

When we compare against other racing games, the key difference we see is that first of all, it is in full 3D (not 2.5D). Moreover, it includes obstacles and roaming enemies - a feature not common in many racing games. If time permits, there will also be bonus features of customisable racecars, characters and tracks.

## Structural Plan

```
|-- Lego Racers Remake
  |-- Obj3D.py
      Object 3D Class: Super class for all more complicated 3D objects
      (virtually everything tbh)
      Handles models and textures, calculations of dimensions, offset
      from center, positioning, collision box drawing etc.
  |
  |-- Racecar.py
      Racecar Class: Subclass of Obj3D.
      Handles racecar with calculations/functions for positioning,
      rotations, speeds and acceleration. Also handles friction and collision
      with Racetrack/walls and floors.
      Handles the passenger too. Purposely made to be scalable for future
      customisability
  |
  |-- Racetrack.py (TODO)
      Racetrack Class: Subclass of Obj3D.
      Handles racetrack by housing collision solids, and is made through
      generation of bricks/walls and a floor
```

```

|
|-- Enemy.py (TODO)
    Enemy Class: Subclass of Obj3D. Contains subclasses of enemy types.
Note that enemies here are more like obstacles and are different from enemy
cars.
    Handles enemy movement, collision detection, AI etc.
|
|-- Powerups.py (TODO)
    Powerup Class: Subclass of Obj3D. Contains subclasses of powerup
types
    Handles positioning, collision detection etc.
|
|-- Game.py
    Main game file that runs the game. Handles timers, collisions,
camera, keyboard events, GUI, audio etc
|
|-- audio
|   |-- backgroundMusic.mp3
|   |-- collisionSFX.mp3
|   |-- engineSFX.mp3
|   |-- menuSFX.mp3
|
|-- models
|   |-- crate.egg
|       Brick for racetrack
|
|   |-- forest.egg
|       Backgrounds
|
|   |-- groundroamer.egg
|   |-- racecar.egg
|       Racecars
|
|   |-- penguin.egg
|   |-- person.egg
|       Passengers
|
|   |-- tex
|       |-- personTexture.png

```

The full file structure of the project can be see in the [Git repository](#).

Project structure with sub-projects can be seen under [Github Projects](#), and is dynamically updated.

## Algorithmic Plan

More details of sub-projects and sub-tasks are found under the [Projects](#) tab.

### Race Car dynamics

#### Collision

- ☒ Collide and slide along wall

Handle with `CollisionPusher`

`CollisionEvent` that fires will also alter the car's speed and acceleration

- ☐ 3D floor handling and collision

Handle with `CollisionRay` and `CollisionFloor`

- ☐ With other cars More complicated; but again handle with `CollisionPusher`

Will require speed and direction to change; trigger through a `CollisionEvent`

Might also trigger other effects that would happen if the player struck an enemy or obstacle (KIV).

## Movement

Car will have movement that includes friction, acceleration and velocity etc:

- ☒ Friction and Front/Back Movement

Friction will be proportional to speed; with acceleration in opposite direction

Pressing arrow keys will increase the acceleration Overall this will give the illusion of acceleration and friction

- ☒ Rotational Movement

Similar to front back movement, except it'll just set a rotational speed, and a negative acceleration, causing the car to turn but then slowly decelerate to a stop

Need to ensure that car will not start rotating unless also moving forward

## Racetrack Generation

Racetrack will be created by generating multiple bricks/crates

- ☐ Walls

All `CollisionSolids` associated with the `Crate/Brick` object into one `CollisionNode`

Helps get rid of issues with car running straight through crate/walls.

- ☐ Floor

Floor needs to have elevation too. Will need `CollisionRay` and `CollisionPlane`, combined with `CollisionHandlerFloor`

- ☐ Representing race track

First test will probably be hardcoded

After this, will most likely read from a file the positions of the walls and floor elevations

## Enemies

2 main types of enemies: roaming enemy (boulder rolling back and forth) and an oscillating enemy (wall moving up and down)

- ☐ Movement

Enemy is a class which contains x and y positions. Use intervals to move the enemy

- ☐ Effects and Collisions

Collision handling will be the same as if the enemies were normal walls; effects will also be the same, ie the player will slow down

## Timeline Plan

1. Get racetrack walls working
2. Get racetrack floor working
3. Fix camera views (easyfix)
4. Add laps
5. Create simple collectable powerups
6. Create moving enemies
7. Misc. add-ons (music, sfx, speedbar, minimap)
8. Enemy cars

More details under [Projects](#) tab.

## Version Control Plan

This [Github repository](#) controls both the code versions, as well as the structure and timeline of the project (see [Projects](#) and [Issues](#)).

Commits automatically reference and close their associated issues.

## Modules Used

[Panda3D Engine](#)

Checkout [requirements.txt](#) to see the external Python libraries used in the repository.