# Quidditch World Cup

Yuqing Guan (yg2392), Jihan Li (jl4346), Yilin Xiong (yx2274)

## Main goal:

Inspired by the world-famous fiction, Harry Potter, we use OpenGL to implement a Quidditch game. A user can select one among the four houses is Hogwarts: Gryffindor, Ravenclaw, Hufflepuff and Slytherin. After selecting one team, our program will randomly assign a rival team controlled by an artificial intelligence.

We simplified the Quidditch game to improve the maneuverability of our game. When playing the game, the user can control a wizard/witch and let it fly by a broom. The players will compete to get a ‘Quaffle’ ball. When one player of the user’s team catches the ball, the user can fly with this ball, pass it to teammates or shoot one of the rival team’s holes to score. One team will get 10 points for each goal, and the first team with 150 points wins.

## Algorithm:

TODO AI

## Results and achievements:

### 3D rendering:

* Background: We use a skybox as the background. If there does not exist a terrain map, our program will randomly generate one terrain and smoothen it by Gaussian and mean filters, otherwise our program will load the existing terrain.
* Stadium: We import a model of a Quidditch stadium, locate it in the center of scene
* Player: We use a model of a wizard with joints and links. The textures of the coat, shirt and emblem of this model are replaced according to the player’s team. We use forward kinematics to change the pose of players during the whole game.
* Collision object: The players and the ball all have a collision radius within which two objects will be detected as collided and then trigger control and AI functions
* Minimap: In order to help users find the positions of players and the ball, we use a minimap to indicate all collision object in this field.
* Models, textures and shaders: We develop loaders to import OBJ and IQE models, including the vertices, UV coordinates, normals, meshes and textured materials. For IQE models, we also import joint and link information. All models are rendered by shaders based on Bling-Phong reflection model.

### Interaction and control:

* On-screen control: We use ‘screen’ objects to manage 3D models and interactive objects.
  + The loading screen will use the progress and logs of the loading process by multi-threading.
  + After loading or exiting from the playing screen, you can see a start screen with two buttons. You can enter a new game or just exit our program.
  + Team selection screen and play screen: In team selection screen, you can choose one of Hogwarts’s four houses as your team and our program will randomly set another house to be the rival team. After selecting your team, you can enter the play screen to play the game. When the game starts or one of the two teams wins, a CG animation will be played.
  + Modal dialogs: When the user requests to exit the program or the current game, a modal dialog will be displayed to ask for the user’s confirmation.
* Mouse control: We provide buttons in the start screen, team selection screen and modal dialogs. Buttons are bind with listeners to perform different actions.
* Keyboard control: A keyboard is the main controller of our game:
  + All-time control:

|  |  |
| --- | --- |
| Key | Effect |
| Q/ESC | Exit game in play screen  Cancel operation in modal dialogs  Exit program in other scenarios |
| F2/F12/P | Screenshot |
| F11 | Toggle full screen |
| Enter | Start new game in start screen  Confirm operation in modal dialogs |

* + In-game control:
    - TODO How to play our game

### Artificial Intelligence:

TODO AI, Please introduce it in detail in the ‘Algorithm’ section and mention it briefly here

## Contributions:

* Jihan Li: Team leader, TODO
* Yuqing Guan: Basic framework of this program, scene, models and collision objects
* Yilin Xiong: TODO

## Notes:

* In order to test our program, please extract ‘Final\_ yg2392\_ jl4346\_ yx2274.zip’, change directory to ‘Quidditch’ and then execute ‘ant’.
* Because we use multiple threads in the loading process of our program, this program cannot be X11 forwarded if it is tested on a remote machine. Therefore, please test our program in local machines.
* Since our program involves the rendering of many complex shapes and textures, it is better to run our program with a high-performance graphics card.

## References:

* TODO