CPSC 565: Emergent Computing, Winter 2020

Assignment 2: Quidditch Simulation (Unity)

Due Date: Friday, March 6, 2020

Quidditch is a competitive sport in the wizarding world of Harry Potter, a series of fantasy novels written by J. K. Rowling. It is a game played by wizards and witches, where there're two teams riding on flying broomsticks trying to score as many points as possible. In this assignment we will implement a simplified version of this game.

The goal of each team will be to touch (collide with) the golden snitch as many times as possible; each time the snitch is touched by a player, his team gains one point, the snitch disappears and re-spawns at a random location instantly. Whenever that happens, there should be a printout of the new score in the console.

The golden snitch will be modeled by a small golden sphere that randomly flies around the Quidditch field. While the snitch should fly around in a random fashion, you should make sure that it doesn't leave the boundaries of the field; and that if it hits the ground, it should fly (bounce) right back.

Each Quidditch player will be modeled by an object that can indicate its direction and is colour coded by team (red for Gryffindor and green for Slytherin). The players should be influenced by two main urges: the urge to catch the snitch, and the urge to avoid frequent collisions with nearby players. You will need to design strategies for the players to make the game interesting. Note that there should be at least 5 players in each team and at most 20, and that each team should have its own single starting point on the ground. If a player hits the ground, he should appear in the starting point of his team, and resume the game.

All collisions in the game obey the law of conservation of momentum, and ignore gravity when a player is flying. In addition to flying, the players have the ability to tackle each other when they collide; this happens according to a predefined probability assigned to each team. When a player is tackled, he should make a free

fall until hitting the ground, appear at his team's starting point, and resume the game.

The two teams have different parameters as the Gryffindor team is known to be faster, and the Slytherin team is known to be better at tackling opponent players. Here's an example set of parameters for each team:

Gryffindor: maxVelocity = 16, maxAcceleration = 20, tacklingProb = 0.3

Slytherin: maxVelocity = 13, maxAcceleration = 17, tacklingProb = 0.7

The tacklingProb parameter defines the success rate in which a player successfully tackles an opponent player when they collide.

The golden snitch has fewer parameters that influence its movement. Here's an example set of parameters for the snitch:

maxVelocity = 16, maxAcceleration = 20

Implementation Requirements:

This assignment must be implemented using Unity 2018 or above (https://unity3d.com/cn/get-unity/download). The game should be easy to observe in the play mode.

You are required to implement the above behaviour and parameters. You should also implement 2 other sets of parameters to reflect your own strategies. Each set of parameters should be implemented as a difference scene.

Documentation:

Describe your strategies, the behavior they created and the effect it had on the score of the game.

Submission:

Export your project as a Unity package file. Submit your Unity package file and additional document using the D2L system under the corresponding entry in Assessments/Dropbox.

Grading:

Your grade will be a letter grade based on the following rubric.

A+	All requirements for an A plus either: UI elements for scoreboard User defined strategies are unusually interesting and well implemented Interesting and smooth camera operation
A	Team players are colour coded and show direction of travel Each team has a unique spawner location that spawns new players Players and snitch exhibit required behavioural urges and limits Players can tackle opponents and tackled players behave appropriately Score updates are shown Velocity, acceleration, and tackle parameters are implemented Scene is easily viewed Good documentation and analysis
A-	One or two minor errors in implementation, or Documentation or analysis not quite adequate

B+	One or two minor errors in implementation, and documentation or analysis not quite adequate, or A significant implementation error but no other problems
B	A significant implementation error as well as one or two minor issues and docs or analysis lacking
B-	A significant implementation error, one or two minor issues and docs or analysis lacking
C+	A little better than a C
C	Multiple errors of note, documentation or analysis completely inadequate or missing, but an effort was made
C-	A little worse than a C
D	Little or no effort made, solution produces some results
F	Submission missing or non-functional