Rugby Roundup
COSC 435 Final Project Proposal
Group Members: Anna Hart, Allegra Knox

Introductory Statement

Our goal is to design and implement a game that represents a hero(character) playing on the rugby 'pitch', or field. The objective is to catch rugby balls falling from the sky while trying to prevent the ball from falling on children who are running around the field.

The game will consist of 7 levels, at the end of each level, the user will choose a new hero from the bench consisting of 6 unique rugby players. Each player will have a different jersey as well as skill set (speed, strength, response time, etc.). A level is passed when a player earns 13 points: 1 point is earned for catching a ball, 2 points are earned for saving a child. A level is failed if a child is hit 3 times with the ball. If a level is failed, the user must repeat it to continue on to another level.

Each level will progressively become more difficult. Elements to add to the difficulty are a more complex path of the child, more children added, increase in ball speed, and increase in number of balls.

The key user interaction is the user controlling the hero's movements along the plane (pitch). The user will also control the hero's ability to push the child, saving them from the ball. The user will also be able to click to choose each player at the beginning of the level.

The hero will be modeled with a series of transformed cubes plus a sphere for a head. The child will use the same model at a smaller scale. The character's physical traits can be stored in an object which will determine their speed, strength, and texture (appearance). The players on the bench will be 2D elements, but each girl will look differently as explained above.

In designing the project, Allegra will design the characters, trees, and uprights (3D modeling) and textures (2D). Anna will handle the game logic and animation. We will collaborate on pulling both aspects into the plane as well as handling collisions, scoring, character changes, as well as any other elements that come up. We also plan on aiding each other when arriving at difficulties in our respective parts.

Technical Outline

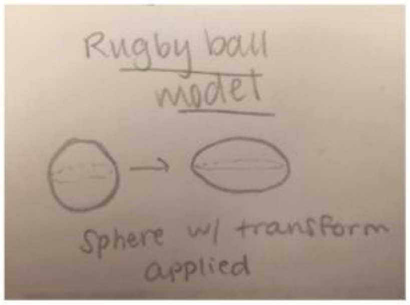
1. **Falling balls:** The balls will fall in a vertical line. Their x and z values will not change as they fall, so only their y values will have to be updated. We will be able to tell if the hero can catch the ball (or if a kid will be hit) by comparing x and z values, then see if the height of the person is high enough to match the falling y value of the ball.
2. **Moving kids**: The kids' y values will not change, however their x and z values will. They will run around the field randomly but adhering to a specific behavior. For example , Kid 1 might have a "running radius" of 10 units but a speed of 10, while Kid 2 will have a

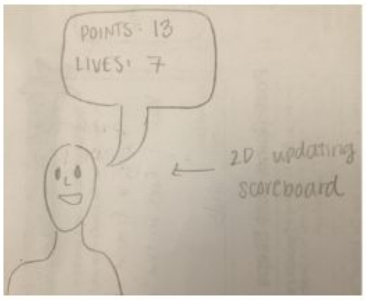
running radius of 20 but a speed of 5. We can implement this by using an animation keyframe track. [3]

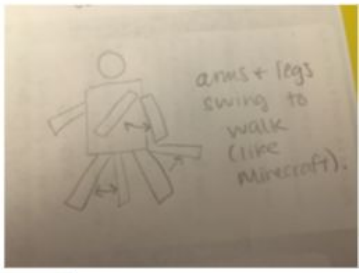
3. **Collisions:** There will be one collison, the hero and the children. The hero can push the kid out of the way of the ball if the collision radius of the hero gets close enough to touch the collision box of the child. We will implement an axis-aligned collision box to do this. [1].
4. **Keyboard Events-** Characters respond to keyboard events [9].
5. **2D updating scoreboard**: We will have a cartoon version of our coach acting as a live-updating scoreboard. We will keep an object containing the scores, and when they get updated (kid gets saved/catch a ball etc) the display will be redrawn. We can use an event dispatcher to alert the 'scoreboard' to changes in the game [4].
6. **3D Plane**: We will have the pitch be a large 3D plane. We can apply a texture to this to give it the color and lines of a rugby pitch.
7. **Physics engine**: implement an animation for when character collides with kid
8. **Clickable characters**: When the player loses, they get to choose the next player off the bench by clicking her.

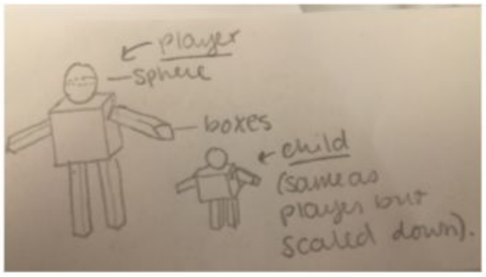
Objectives

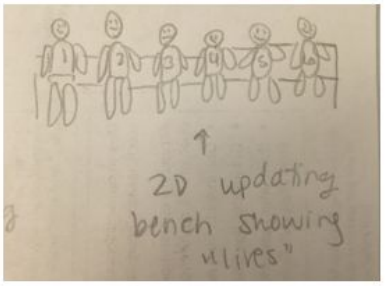
1. **Scalable modeling with toon shading-** Model hero and characters in a scalable design that allows for use of different textures and features. Use toon shading to a cartoon like style. [5]
2. **Shadows** - Model shadows of falling balls using shadow mesh and shadow material from THREE.js [6]
3. **Animation-** implement animation for game players to look realistic in actions. We can implement this by using an animated model with controls [8].
4. **Plane-** The plane will be up close to the near-z of the field of view. This will make it appear that the field takes up the whole horizontal view of the screen, when in reality it is not an infinite plane. The field of view will be fixed, so when the character moves around, the pitch stays in place.
5. **3D Background**- randomly generate a treeline (in addition to our skybox) to give the horizon/background a little bit more interest.
6. **Texture-** Apply textures to field, ball, and character models. Implement different textures for different characters.
7. **Skybox**: include a skybox with an image of what you would see from the field using a cube-map texture and a 360 image. [2]

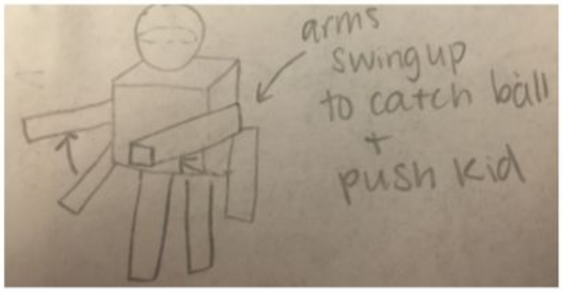


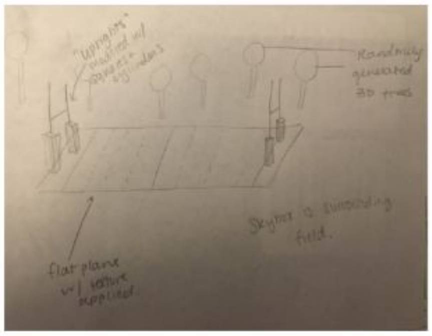











"uprights" modeled w/ spheres + cylinders
Randomly generated 3D trees
flat plane w/ texture applied
Skybox is surrounding field.

Sources

1. https://learnopengl.com/In-Practice/2D-Game/Collisions/Collision-detection
2. http://ogldev.atspace.co.uk/www/tutorial25/tutorial25.html
3. https://threejs.org/docs/index.html#api/en/animation/tracks/VectorKeyframeTrack
4. https://threejs.org/docs/index.html#api/en/core/EventDispatcher
5. https://threejs.org/docs/#api/en/materials/MeshToonMaterial
6. https://threejs.org/examples/?q=shadow#webgl_shadowmesh
7. http://guillaumeblanc.github.io/ozz-animation/samples/blend/
8. https://stemkoski.github.io/Three.js/Model-Animation-Control.html
9. https://stemkoski.github.io/Three.js/Keyboard.html