**Game Explanation**

It is a first-person puzzle game that uses the colour to apply objects with different effects. There are 7 different colours each with their own effects. Applying multiple colours to an object will generate a new effect. The game is level-based where they must solve a set of puzzles in order to progress through the game.

**Game Mechanics**

**Modelled Objects and Entities**

The game is an amalgamation of the games Portal and Antichamber where the design of some the objects are meant to be very Sci-Fi as it fits with the whole theme of the game that being using science to solve puzzle. The modelled objects are designed simpler to maintain consistency of the overall game style. Since the game is designed similar to how Portal does it, the modelled objects are made to be something similar to Portal as well.

**Graphics Pipeline and Camera Motion**

**Shaders**

Several shaders are used in the making of this game,

* Dissolve shader, which is used to allow the player to progress through the game either done by rendering a pathway or de-rendering a blocked path for the player to traverse through. The shader revolves around the built-in function clip() where it won’t render certain pixels if the specified value is less than 0. The shader uses the local position of the object to identify the centre of the object where the clipping will start and end. Adding a noise texture will randomize which pixels will be rendered and which aren’t, this gives a dissolving look. Float values are used to increase the level of randomness.

* Hologram shader, which is used for aesthetics. The shader revolves around rendering or not rendering colour of a pixel. This is done using a combination of sin, cos, and max function. If the sin or cos function of the world position of the pixel is negative, then using the max function it will assign the value 0 onto the colour of that pixel. To generate the hologram look, the shader will have the culling turned off which allows the colour to be seen in the inside of the object.
* The Fresnel/rim shader, which is also used for aesthetics. This is done by manipulating the colour’s alpha depending on the pixel’s vertex. The shader will display a high alpha value on the rim of the object and that value will gradually decrease the closer the pixels are to the centre of the object.
* The paintball shader uses a pre-generated Perlin noise texture to vary the displacement of the vertices on the paintball sphere. Using the \_Time.y function to vary displacement by time, and the sin() function to oscillate the displacement, the vertices oscillate in the direction of the normal, giving an illusion of a watery sphere.
* The outline shader requires the camera to obtain the depths and normal of the pixels on screen. By using DepthTextureMode on the camera, this allows the material with the outline shader to access the depths and normal from the camera. From here, for each vertex

**Querying and Observational Methods**

**Changes after feedback**

**Sourced code/APIs**

* The idea around the code for the player movement is based on this website,

<https://medium.com/ironequal/unity-character-controller-vs-rigidbody-a1e243591483>

**Group Members**

* Nicola Halim
  + Made the camera and player movement, including jumping and running
  + Made the dissolve, hologram, and Fresnel/rim shader using unlit shader
  + Made the colour effects through the use of inheritance
  + Made the models of all the objects
  + Made the script to allow the player to pick up and drop objects.
* Samuel Tumewa
  + Made the paintball shader with Perlin noise and the edge detection outline effect shader
  + Made the tutorial levels
  + Made paintball shooting and object coloration mechanism
  + Made the level instruction textures
* Marcella Yoesman