CS 408
Prof. Alain Crotte

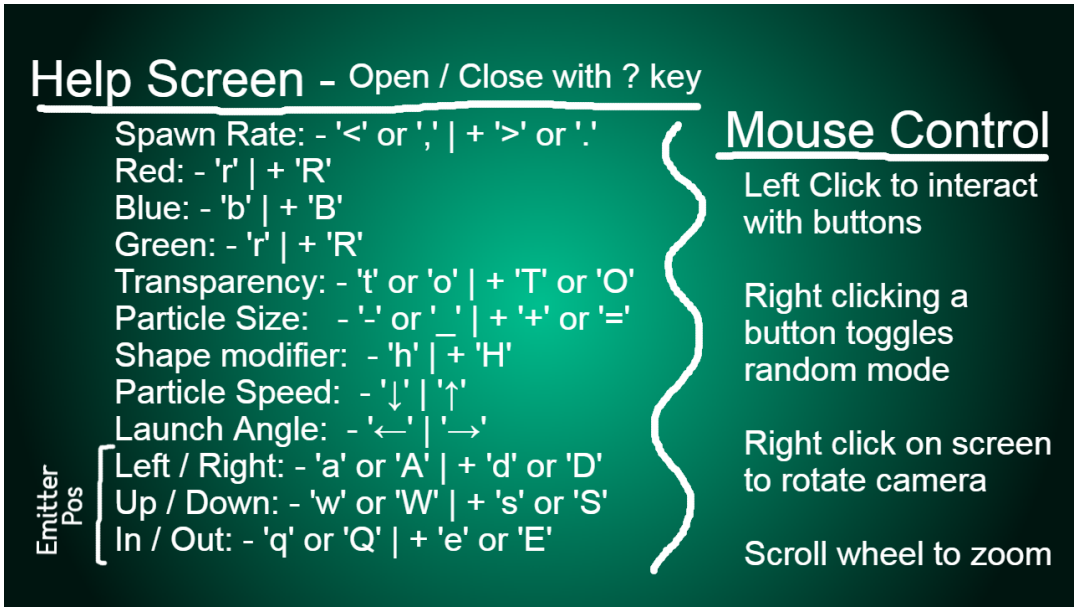
Ariston Bodnarchuk
200285478

Assignment 1 - 3D Particle System

All of this project can be seen on my github, please email me at ***bodnarca@uregina.ca*** and provide an email address so I can add you as a collaborator to view my project.

User Guide

When launching my projectile system the user will be met with an instruction screen; The user cannot use the program until they have closed off the instruction screen which requires minal reading of the instructions.



Keyboard Controls

- Open / Close with ? key
- Spawn Rate: - '<' or ',' | + '>' or '.'
- Red: - 'r' | + 'R'
- Blue: - 'b' | + 'B'
- Green: - 'r' | + 'R'
- Transparency: - 't' or 'o' | + 'T' or 'O'
- Particle Size: - '-' or '_' | + '+' or '='
- Shape modifier: - 'h' | + 'H'
- Particle Speed: - '↓' | '↑'
- Launch Angle: - '←' | '→'
- Left / Right: - 'a' or 'A' | + 'd' or 'D'
- Up / Down: - 'w' or 'W' | + 's' or 'S'
- In / Out: - 'q' or 'Q' | + 'e' or 'E'

Mouse Controls

- Left Click to interact with bottom row buttons and increment / decrement specific parameter
- Right click to either
  - Toggle random mode on a particle parameter input via respective button
  - Rotate Camera along Y axis
- Scroll Wheel to adjust perspective zoom

CS 408
Prof. Alain Crotte

Ariston Bodnarchuk
200285478

Design

Processing Overview
Processing uses a setup() and draw() function; setup is run once before the draw loop starts establishing window parameters (framerate, size, etc) and initializing variables; after setup completes the draw loop is run a certain amount of times a second (default 30) defined in setup. Running processing is as simple as downloading the right version for your system and exporting the zip file.

Program
My draw loop acts as a driver for my particle system. Having made 2D particle systems in the past I followed the same design processes I used previously. I use three classes, although one class is built entirely to assist with keyboard and mouse functionality and is not strictly necessary for the function of the projectile system. The three classes are as follows: **ParticleSystem**, **Particle**, **ButtonData**.

The **Particle system** manages all of the particles that are on screen, tracks the user-provided parameters for the particles, adds new particles, and handles the majority of keyboard and mouse input. Some functionality for mouse and keyboard had to be handled outside of the class for simplicity as processing contains some predefined keyboard and mouse events that make the interaction more straightforward to implement through external functions that feed into the particle system object. The particle system tracks the particles through an array list of type particle that is iterated through in reverse to simplify handling deceased particle objects as if iterated from front to back, removing an object would change the index and potentially lead to index out of bound exceptions which is extremely unwanted. It tracks on screen buttons through a strip of onscreen buttons located at the bottom of the screen that track both keyboard and mouse functionality and relay the user input back to the particle system class which is then fed into particles that are being generated without affecting the preexisting ones. This is accomplished by making the particle class as the creation of new particles should not affect the old ones. The particle system also manages the shapes of the particles as upon creating a particle system object it will generate all possible shapes (from the range of shape modifier values) and store them into an array to pass to the particle constructor to improve performance of the program (speed) instead of drawing each complex 3d shape each time. The particle system accomplishes this via two functions, and storing the corners of the cube, and passing the corners to a function that constructs one face of the “cube” which is called from a function that builds a specific shape modifier of the “cube” and pieces together each of the 6 faces into one single “cube” esque object referred to as a PCube in my code to play along with processings naming conventions. I have previously made custom shapes in 2D but this was my first endeavor into 3D with processing and creating custom 3D objects. I did not want to create 3D models to feed into the program as it would have damaged my pride as a programmer. Because the particle class is immutable it is not possible for the particle system class to change

the value, however it may not be considered truly immutable as values within the particle objects update internally. The particle system also manages the “camera angle” which is technically just a rotational matrix manipulation applied to everything drawn during the particle system display function; this works well but has the added effect of moving all the particles when the origin is changed. Although this goes against each particle not being affected by the other, I am alright with leaving it in as it saves a lot of memory by not having to store a separate origin along with the position vector in each particle which is 3 less float values needing to be stored. I could come up with a complex thing that uses one value to get a percentage of screen that it is at, but really just leaving it as it makes the most sense for what should be a straight forward assignment. The final thing the particle system class manages is the help screen that is displayed on launch and toggled with the question mark key (not case sensitive). This also acts as a pause feature because particles are immutable and the particle system does not run updates if the help screen is up to save on performance as what is the point of running a particle system that can’t be seen.

The **particle** class is immutable as the particles are not supposed to be influenced by one another and instead be generated separately, where a new particle does not change the existing ones. The particle is passed its shape by the particle system, as the particle system has pre-rendered every 3D object for possible shape modifiers to cut back on rendering and improve performance. Particles have a position to track where they are in relation to the origin of the particle system, and have additional vectors to track velocity, spin, and gravity. I felt adding a random spin to each particle, along with some gravity and air friction gave the particle system a much better feel and made it more enjoyable overall as the particles don’t just fly off in a straight line but instead have different arcs. Gravity is not a constant but rather based on size of the particle so that the bigger shapes fall faster than smaller shapes. This gave more life to the simulation. Particles also track their color, shape, size, death timer, and whether they are alive, as well as a random spin direction. The particle class uses three different constructors based on which parameters are set to random mode in the particle system. One constructor is completely random and was the original random constructor the entire system was tested on. The second is a constructor where just the velocity is random, this allows for particles to gain a z element randomness that usually cannot be controlled via user input. The third is a full parameter constructor where all values are passed in.

There is not much to say about the **ButtonData** class. It could technically be a multidimensional array instead. Originally I wanted the button class to be more robust and handle all the functionality of the buttons but it was simpler to make it handle only the visuals and data of the buttons but leave the functionality up to the particle system. It was easier to implement this way as otherwise I wasn’t exactly sure how to pass particle parameters back and forth as well as different triggers, or multiple ways of triggering buttons.

CS 408
Prof. Alain Crotte

Ariston Bodnarchuk
200285478

Creative Features

Admittedly I have lost track of the creative features, or what even constitutes them. When the assignment was first presented we were told it must be 3D, after I had asked haha, so I wrote a 3D particle system in processing then left the documentation for later while I worked on different assignments and for a submission page for the assignment. In one of our classes however some students asked again and 2D became acceptable. So this is basically just a lot of words to say. I have no idea if 3D counts as a creative feature or not lol.

Most or all of the creative features have been mentioned or talked about in some way by this point so I will quickly summarize. Keyboard functionality is represented on screen with buttons that are also clickable. Particle spawn rate can be adjusted with the '<' and '>' keys but is not reflected in the onscreen buttons as it wasn't a requirement and I didn't want to clutter the screen and make buttons difficult to read. Particles spawn is limited by performance, the first check for a new particle is if FPS is at least 50%, then it is determined by spawn rate. Spawn rate starts at 1 particle a second but can be increased to 1 a frame or 60 particles a second, but one particle a second feels better as it is better to start the simulation off not rendering much instead of full send in my opinion. Performance is also displayed in the console. Along with keyboard functions there are also mouse functions. Each on screen button can be clicked to adjust, scroll wheel zooms in and out of the particle system, right clicking will either rotate the camera or switch a button to random mode. There is a help screen that displays controls and doubles as a pause button. Particles generate with a random spin, in a random direction, a gravity based on size, and an initial velocity. Buttons on screen for color, unless set to random (to prevent it from looking seizure inducing at higher spawn rates), will match the color of the particles being spawned.

Unintended "Features" (Bugs)

This issue does not exist when using processing regularly (2D rendered do not have this issue) but there is a reported bug for processing with the 3D renderer that prevents the maximize button from being disabled that is specific to the P3D renderer.

Moving particle origin moves all particles, this was mentioned earlier but basically I left this in as it saves on a lot of memory given that this would require each particle to track its origin alongside its other variables. It probably wouldn't make the biggest difference but given it is in 3D with a lot of matrix manipulations happening already it doesn't hurt in my opinion.

Particles can fly in front of the buttons due to depth. I think it actually is a cool effect and ended up leaving it as is because it really amplifies the 3D feel to the simulation