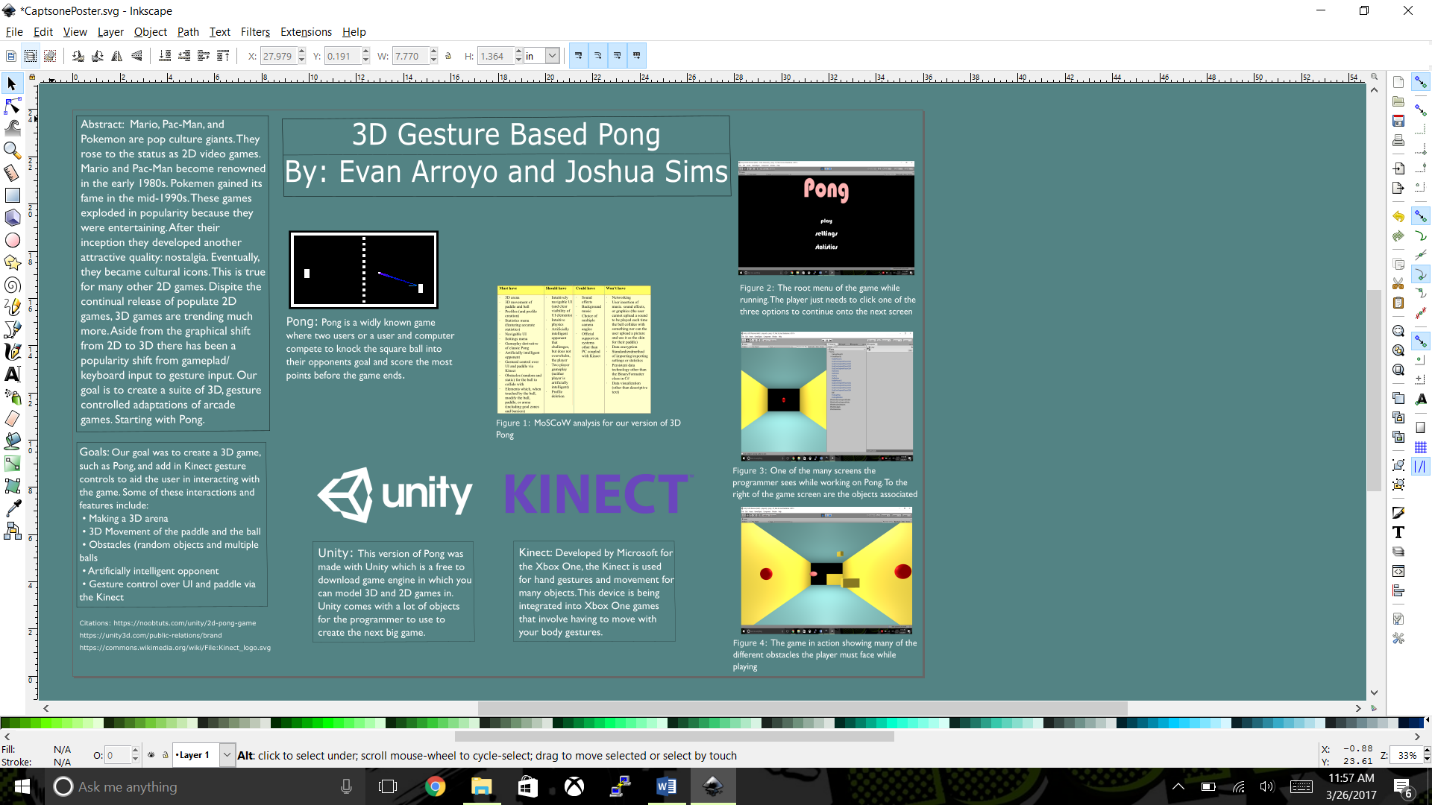
Sprint 11 Report

by Evan Arroyo and Joshua Sims

The goals set for this sprint were to create a final draft of a poster which describes this project, to fix the excessive spawning of extra balls, and refactor the gameplay effects such that they are introduced as the player’s score increases.

We completed a final draft of a poster which describes this project. The poster includes an abstract, screenshots, and information regarding our project and the tools which we’ve used to develop our project. A screenshot of that poster appears below this paragraph.



We fixed the excessive spawning of extra balls. We discovered that the issue arose from multiple, simultaneous calls to the OnCollisionEnter function. When a ball hits the goal zone such that it hits the border between two or more goal zone segments, the ball’s OnCollisionEnter function is called for each goal zone segment that meets at that border. Upon scoring a point, the extra balls were programmed to decrement an integer field which represents the current number of extra balls. So that decrement function was called for each goal zone segment that the ball collided with and thus when an extra ball hit a border between two or more goal zone segments, it overdecremented the number of extra balls in play. That overdecrement then caused the EffectExtraBalls class to spawn more extra balls than it was intended to because EffectExtraBalls spawns a random number of extra balls as long as that number will not cause the number of extra balls in play to exceed the maximum allowable number of extra balls (which is set to 4). We fixed the excessive spawning issue by refactoring the extra balls effect such that each extra ball changed a boolean (which represents whether that extra ball is active or not) to false when it reaches a goal zone segment – there is no way that an extra ball can overdo changing its boolean to false.

We refactored the gameplay effects such that they are introduced as the player’s score increases. There exists a boolean field for the Behaviour class called enabled; the enabled field determines whether the update function is called for that object. Each of the effect classes (our classes which control the additive gameplay effects) is a subclass of the MonoBehaviour class which is a subclass of the Behaviour class. Thus, we simply programmed our ScoreKeeper object to enable each effect class after player one reached a certain score.