**Create – Applications From Ideas  
Written Response Submission Template**

Please see [Assessment Overview and Performance Task Directions for Student](https://apcentral.collegeboard.org/pdf/ap-csp-student-task-directions.pdf?course=ap-computer-science-principles) for the task directions and recommended word counts.

**Program Purpose and Development**

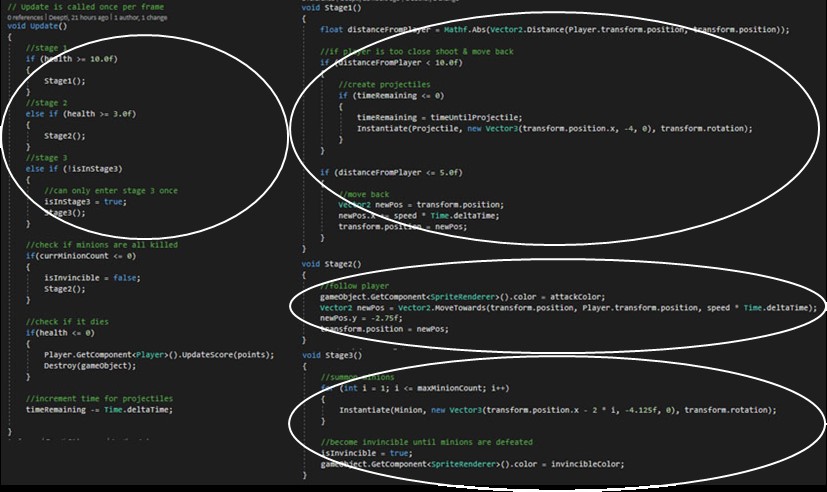
2a)

|  |
| --- |
| This game was programmed in C# with the purpose to entertain people. The video shows a playthrough of the game ending in victory. The player controls the white cube and the red cubes are the enemy. The video shows the player using its attack and defeating various enemies of different difficulties before defeating the boss. The boss has three phases: long-range attacks, short-range attacks, and a final stage where the player needs to defeat summoned mini-enemies. After defeating the boss, the player moved into the green goal area, and the win screen was shown. |

2b)

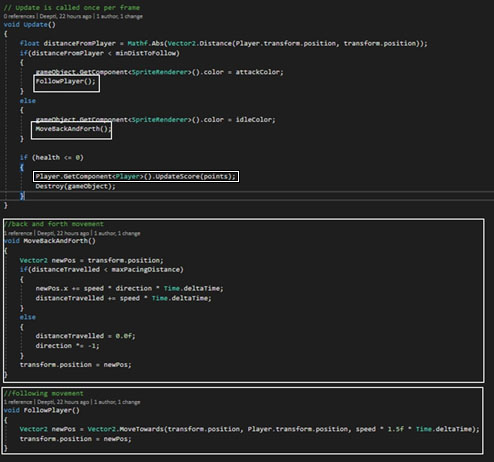
|  |
| --- |
| I worked independently to make a side-scrolling game. I began by creating classes that would control the movement, attacks, and health of the player, basic enemies, boss, and mini-enemies. For the boss, I wanted a multi-stage fight, so I divided the code into three parts that'd run based on the it's health level. I also created a program to generate enemies every few seconds. Next, I created a class that would cause the UI/camera to follow the player.  One difficulty I found when programming this game was with the projectile system. I programmed the boss to instantiate projectiles when it entered a certain phase, but realized that the projectiles were falling due to physics, rather than tracking the player like I wanted them to. To solve this problem, I reset the z-coordinate of the projectile every frame, and added a component to turn off physics for that object.  Another difficulty was having the player check for the conditions of various enemies before attacking. The boss, for example, has an invincibility phase, while the basic enemies do not. To solve this problem, I added an if statement in the program to check which enemy was being attacked. |

2c)



|  |
| --- |
| The first part of the algorithm is a series of if statements that check the boss's current health. If the health is greater than 10, the boss enters stage 1, if the health is between 3 and 10, the boss enters stage 2, and if the health drops below 3, the boss enters stage 3. However, since stage three involves the boss summoning minions, it can only enter it once, so there is an additional check to prevent the boss from entering it multiple times.  This algorithm then uses three additional algorithms detailling what to do in stages 1, 2, and 3.  In stage 1, the program calculates the distance between the enemy and the player using vector math. If it is between 5 and 10, the boss shoots projectiles, and if is is less than 5, the boss moves away from the player.  In stage 2, the boss changes color and follows the player. If it collides with the player, it does damage.  In stage 3, the boss creates minions and becomes invincible, changing its color accordingly. Once the minions are killed, the boss goes back to its stage 2 behavior, and can be damaged. |

2d)



|  |
| --- |
| This abstraction is used to simplify the enemy's actions into smaller tasks. The enemy has three options: moving back/forth, chasing/attacking the player, or dying. Rather than writing both movement scripts in the Update() function, the program was split into the MoveBackAndForth() function and the FollowPlayer() function to simplify it. This way, the code is easier to follow and understand. Another abstraction is the UpdateScore() function. UpdateScore() is a function in the player class, and this abstraction allows the UI to be updated without having many references to the UI. Rather, it is only getting updated in one function in the player class, and that function is getting accessed in many areas. This abstraction also makes the code easier to read, and also prevents the UI from getting updated more than once every frame. |