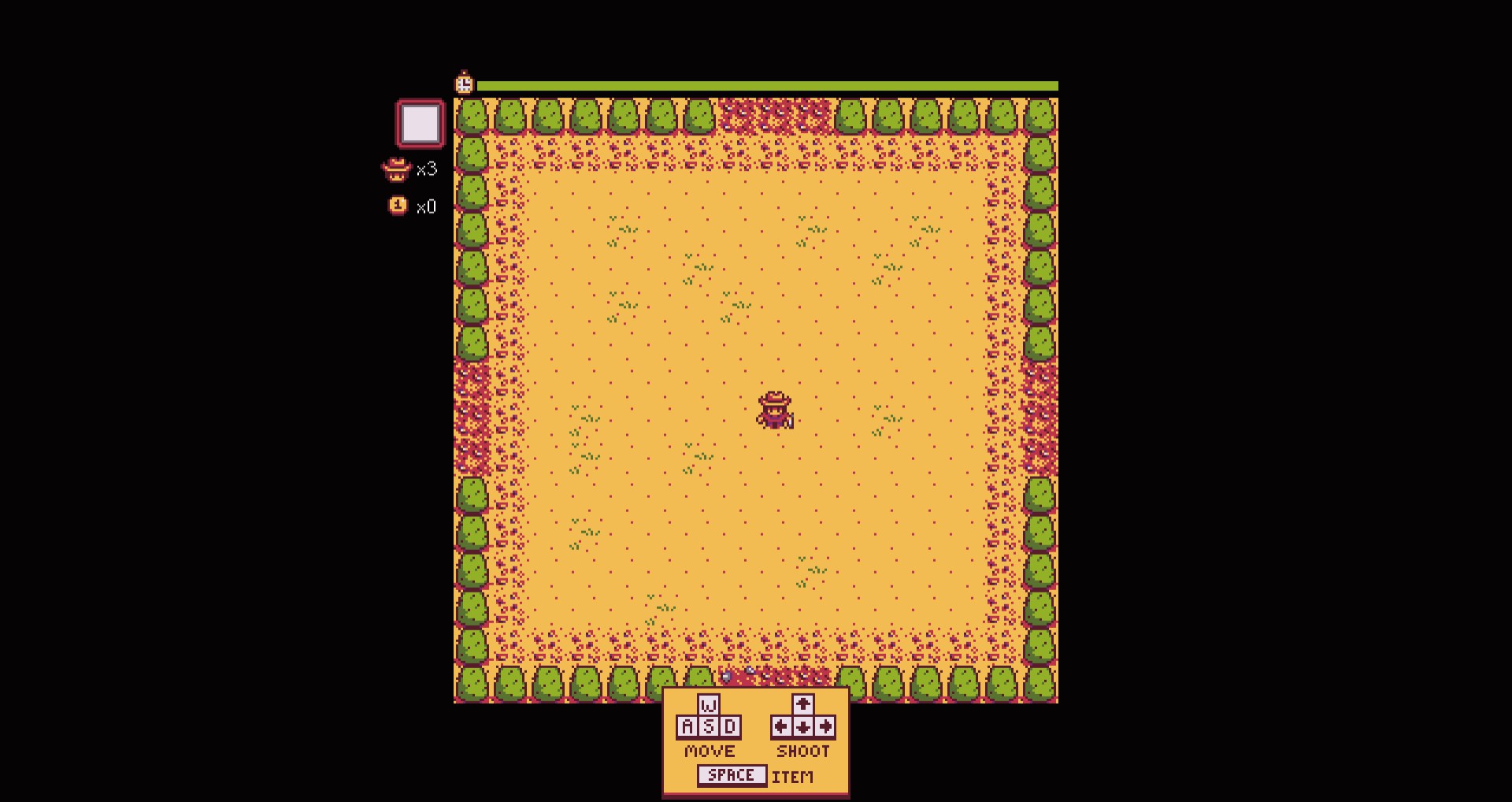
Detailed Game Specification:  
Showdown!

*Course: COMP 2659, Winter 2024  
Instructor: Marc Schroeder  
  
Author(s): Brooklyn Thomson & Adam Cole  
Last Modified: April 8, 2024*

# **1. General Game Overview**

Our game, **Showdown!**, is a top-down shooter where you play as a cowboy or cowgirl who is being overrun by giant snakes from all sides and must defend themself. A top-down shooter is a type of game where the perspective is a bird’s-eye view of the player and play area where you can shoot in many directions to attack incoming enemies. The enemies will come in timed waves with very little reprieve time to regroup between each wave. The gameplay is going to be fast and hectic where the play must keep moving and shooting to avoid getting surrounded.



Barone, E. (2016). Stardew Valley. (PC) [Video game]. ConcernedApe.

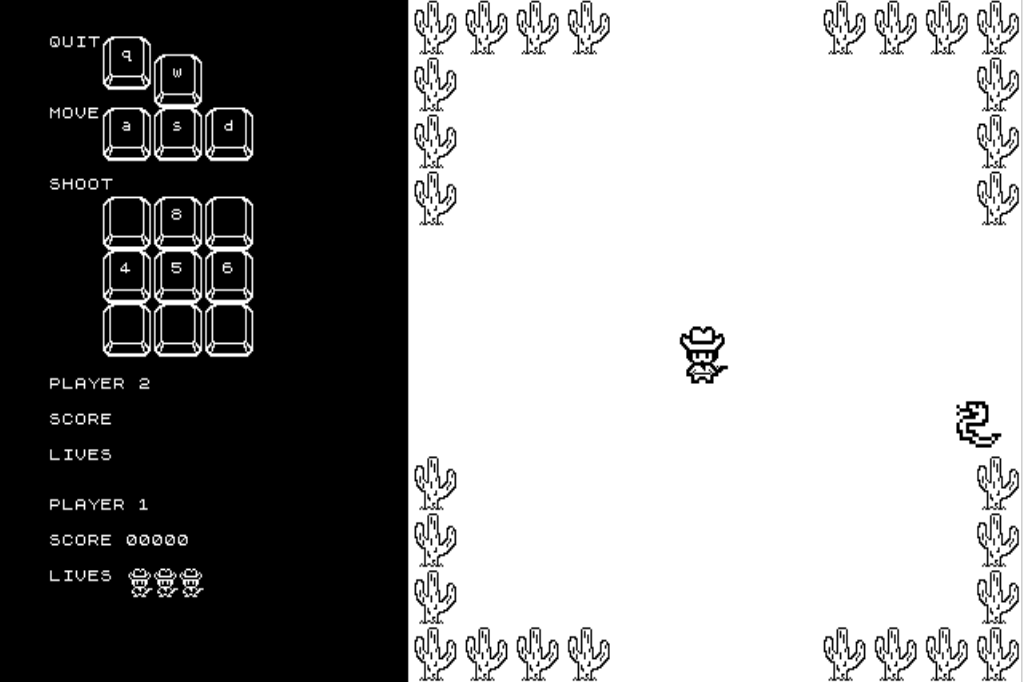
# **2. Game Play Details for Core 1-Player Version**

## **Objectives and Rules**

The starting state of the game has the player in the center of the screen awaiting the first wave of enemy snakes. The score at the bottom left of the screen is set to 0.

You are the hero shooting incoming enemies. The player uses inputs to move their character around the screen and to shoot at oncoming waves of snakes. The player has unlimited ammo. When one of your bullets hits an enemy, it is defeated (blipping off the screen with a death sound), and a point is gained. When a wave is defeated, there will be a brief pause in enemy spawning, then the next wave of snakes will begin.

If the hero comes in contact with a snake, they lose a life, then they are moved back to the center of the screen to resume fighting the same wave of snakes. The initial version of the game will have the player continue until all lives are lost.



## **Objects**

| Object or Object Type Name | Properties | Behaviours | Graphical Image |
| --- | --- | --- | --- |
| Cowboy | * Vertical direction (integer) * Horizontal direction (integer) * Position (integer pair: x, y) * Size (constant integer pair: height, width) * Horizontal velocity (integer) * Vertical velocity (integer) * isMoving (boolean) initially set to FALSE * isFiring (boolean) initially set to FALSE * Vertical firing direction (integer) * Horizontal firing direction (integer) | * Moves * Shoots at enemies * Collide (with enemy) |  |
| Snake | * Vertical direction (integer) * Horizontal direction (integer) * Position (integer pair: x, y) * Size (constant integer pair: height, width) * Horizontal velocity (integer) * Vertical velocity (integer) * Target: cowboy’s position (integer pair: x,y) * isDead (boolean) initially set to FALSE | * Move * Collide (with player) * Disappears from play after collision with bullet |  |
| Bullet | * Vertical direction (integer) * Horizontal direction (integer) * Position (integer pair: x, y) * Size (constant integer pair: height, width) * Horizontal velocity (integer) * Vertical velocity (integer) | * Move * Collide (with enemy) * Disappears from play after collision or leaving the screen |  |
| Lives | * Position (constant integer pair: fixed x, y) * Size (constant integer pair: height, width) * value (integer) | * Updated after taking damage |  |
| Score | * Position (constant integer pair: fixed x, y) * Size (constant integer pair: height, width) * value (integer) | * Updated after each enemy kill |  |
| Map Tiles | * Position (constant integer pair: fixed x, y) * Size (constant integer pair: height, width) | * Static * Barriers or obstacles that prevent movement |  |

## **Physics**

## **Bullet Movement and Enemy Collisions**

The bullet is an object that is 4 pixels wide and 4 pixels high. Pressing the arrow keys will shoot the bullets out in the direction inputted. Bullets can be shot in all directions of the screen. The player has an infinite amount of bullets to use, 30 are in the array being recycled since only so many bullets can fit on screen during play. The bullets move continuously unless they collide with an enemy. Once a bullet exits the screen, it ceases to exist. In the event of a collision between the bullet and the enemy, the score will be updated, increasing it by one hundred points.

## **Cowboy and Enemy Movement**

The cowboy is an object that is 32 pixels wide and 32 pixels high. The character moves in all directions on the screen. The cowboy is controlled by the human player and is moved using the ‘WASD’ keys. The cowboy will remain still as long as no key is being pressed and will move in the direction of a key pressed. The enemy is an object that is 32 pixels wide and 32 pixels high. The enemies are controlled by the computer. The computer tracks the player, following the player's current coordinates. The player cannot exit the screen boundaries.

## **Cowboy and Enemy Collisions**

A collision between the cowboy and the enemy will cause the player to lose a life. If the player is on their last life, then another collision will cause a Game Over.

## **Asynchronous (Input) Events**

| Event Name | Triggering Input Event | Description |
| --- | --- | --- |
| Move up request | ‘W’ key is depressed | Will set the player’s character vertical direction to -1 (upward). Sets the cowboy’s isMoving boolean to “TRUE” and moves the character in the designated direction at a rate specified in the synchronous ‘Cowboy movement’ event. |
| Move down request | ‘S’ key is depressed | Will set the player’s character vertical direction to 1 (downward). Sets the cowboy’s isMoving boolean to “TRUE” and moves the character in the designated direction at a rate specified in the synchronous ‘Cowboy movement’ event. |
| Move left request | ‘A’ key is depressed | Will set the player’s character horizontal direction to -1 (leftward). Sets the cowboy’s isMoving boolean to “TRUE” and moves the character in the designated direction at a rate specified in the synchronous ‘Cowboy movement’ event. |
| Move right request | ‘D’ key is depressed | Will set the player’s character horizontal direction to 1 (rightward). Sets the cowboy’s isMoving boolean to “TRUE” and moves the character in the designated direction at a rate specified in the synchronous ‘Cowboy movement’ event. |
| Cease movement | Releasing all movement keys (or when no movement keys are depressed) | When no key is being depressed the cowboy’s isMoving boolean is set back to ‘FALSE’ (the start state), and the character does not move. |
| Cease upward movement | Release of ‘W’ key | Will set the player’s character vertical direction to -1/1 depending on the last key released. |
| Cease downward movement | Release of ‘S’ key | Will set the player’s character vertical direction to -1/1 depending on the last key released. |
| Cease left movement | Release of ‘A’ key | Will set the player’s character horizontal direction to -1/1 depending on the last key released. |
| Cease right movement | Release of ‘D’ key | Will set the player’s character horizontal direction to -1/1 depending on the last key released. |
| Shoot the gun up request | Up arrow key (‘8’) is depressed | Will set the vertical firing direction of the bullets to -1 (upward). Sets the isFiring boolean to ‘TRUE’ and fires in the designated direction at a rate specified in the synchronous ‘Bullet firing’ event. |
| Shoot the gun down request | Down arrow key (‘5’) is depressed | Will set the vertical firing direction of the bullets to 1 (downward). Sets the cowboy’s isFiring boolean to ‘TRUE’ and fires in the designated direction at a rate specified in the synchronous ‘Bullet firing’ event. |
| Shoot the gun left request | Left arrow key (‘4’) is depressed | Will set the horizontal firing direction of the bullets to -1 (leftward). Sets the cowboy’s isFiring boolean to ‘TRUE’ and fires in the designated direction at a rate specified in the synchronous ‘Bullet firing’ event. |
| Shoot the gun right request | Right arrow key (‘6’) is depressed | Will set the horizontal firing direction of the bullets to 1 (rightward). Sets the cowboy’s isFiring boolean to ‘TRUE’ and fires in the designated direction at a rate specified in the synchronous ‘Bullet firing’ event. |
| Cease firing | Releasing all firing keys (or when no firing keys are depressed) | When no key is being depressed the isFiring boolean is set back to ‘FALSE’ (the start state), and the character does not shoot. |
| Cease upward firing | Release of up arrow key (‘8’) | Will set the player’s firing vertical direction to -1/1 depending on the last key released. |
| Cease downward firing | Release of down arrow key(‘5’) | Will set the player’s firing vertical direction to -1/1 depending on the last key released. |
| Cease left firing | Release of left arrow key (‘4’) | Will set the player’s firing horizontal direction to -1/1 depending on the last key released. |
| Cease right firing | Release of right arrow key (‘6’) | Will set the player’s firing horizontal direction to -1/1 depending on the last key released. |
| Quit | ‘Q’ key is depressed | Will set the quit flag to TRUE. |

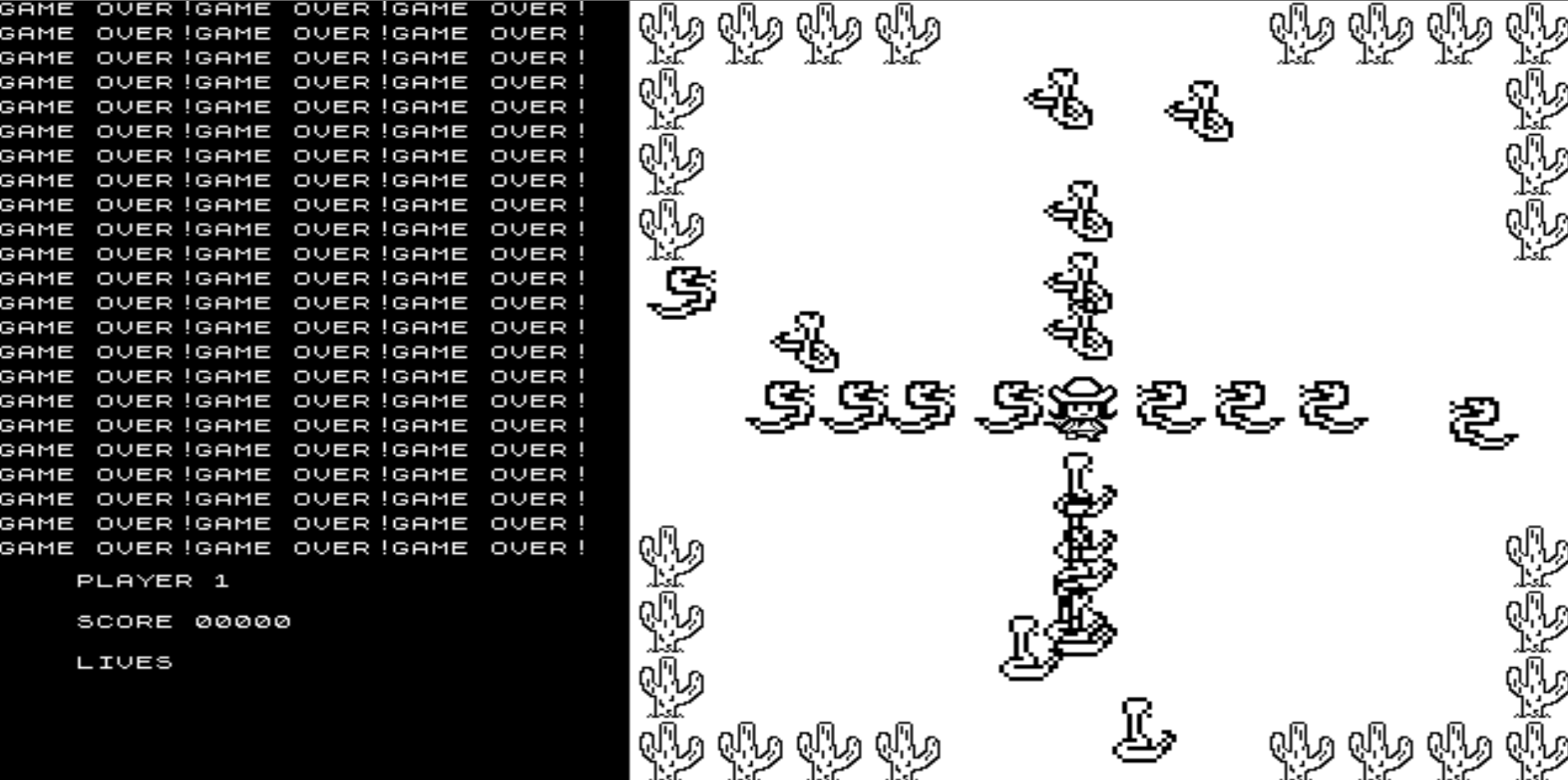
## **Synchronous (Timed) Events**

| Event Name | Trigger Timing | Description |
| --- | --- | --- |
| Bullet firing | Every 1/70th of a second | When the isFiring boolean is “TRUE”, the avatar fires bullets in the direction designated by the arrow keys. |
| Bullet movement | Every 1/70th of a second | Bullets move horizontally and vertically. This will update the x,y position properties in the bullet object. |
| Cowboy movement | Every 1/70th of a  second | When the cowboy’s isMoving boolean is “TRUE”, the player’s character x,y position property will be incremented horizontally/vertically depending on the direction buttons pressed. After updating the player’s character position properties the player’s character horizontal/vertical direction will be set to zero. |
| Enemy movement | Every 1/70th of a second | The enemies will track the player and move towards them. We believe this will be done by the enemy object knowing the coordinate position of the character as they move and moving toward that point. The movement itself will be based on the difference between their vertical and horizontal position compared to the cowboy’s. |
| Enemy spawning | Every second | During a wave (30 seconds), every second, a new enemy will randomly appear at one of the four edges of the screen. |
| Update enemy’s target | Every 1/70th second | The enemies on screen will update their intended target by copying the cowboy's position into their ‘target’ property. |

\*Trigger timing events to be adjusted and refined as needed during the playtesting phase.

## **Condition-Based (Cascaded) Events**

| Event Name | Triggering Condition | Description |
| --- | --- | --- |
| Enemy death | Bullet impact enemy sprite’s bounding box | When a bullet collides with a snake’s hitbox, it will blip out of existence and the appropriate points will be added to the player score(*see score increase*)**.** |
| Score increase | Enemy death | When an enemy dies the score will be increased. |
| Player death | Enemy impacts player sprite’s bounding box | If an enemy’s hitbox overlaps the player’s hitbox, this results in the player dying, losing a life, and then respawning in the center (*see player respawn*). |
| Player respawn | After player death | The player will respawn in the center of the play area. |
| Wave bonus | After Wave | Increases score by 1000 at the end of a wave. |
| Game Over | Loss of all lives | If the amount of lives is equal to zero the terminate game flag will be set to true and a game over screen descends like a curtain, but keeps the final score on displayed. |

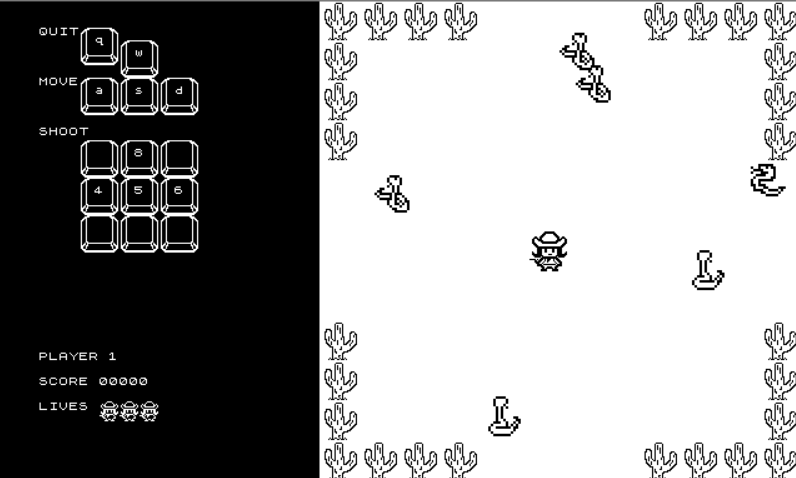


## **Hypothetical Gaming Session**

The starting state of the game has the player in the center of the screen awaiting the first wave of enemy snakes, with both the isFiring and isMoving booleans set to ‘FALSE’. The score at the bottom left of the screen begins at 0. During each wave, enemies will randomly spawn from one of the four screen edges every second for 30 second waves. As the enemies begin to appear on the screen, the player frantically moves into position to take them out with shots from their pistol. Loaded with unlimited ammo, the player continues to kill enemies until the enemy wave is concluded or they’ve lost all of their lives. This could take many waves with the scoreboard increasing as they kill enemies. Each time a player loses a life, they reappear in their starting location in the center during the same wave. Once the player loses all their live ‘GAMEOVER” is printed to the screen.

**3. Game Play Details for Core 2-Player Version**  
In the two player mode, both players are controlling separate avatars on screen at the same time on separate computers (one a cowboy, and the other a cowgirl). They will start back to back in the center, awaiting the first wave of enemies. This way they can act as a team against the enemy waves. The competition between them is whoever can achieve the highest score during their playthrough. Once both have lost all of their lives, the game over screen appears but this time the final scores are compared, and the winning player number and score is displayed on the screen before the offer to play again.

A key component of this mode is designing two sprites that can be easily identified as player 1 or player 2 during play, although the sprites not being so easily distinguishable could be part of the challenge of the 2-player mode as you need to keep track of which avatar is yours. All other rules, gameplay, and graphics will remain the same.



# **4. Sound Effects**

| Sound Effect Name | Brief Description | Event which Triggers Playback |
| --- | --- | --- |
| Enemy death | Short *thunk* sound | Bullet hitting enemy |
| Enemy Spawn | Rattlesound | Every pass of a second |
| Bullet firing | 8-bit style *popping* | Input of arrow key |

Our goal for the background music will be simple but catchy. Something that won’t get too repetitive while playing. It could be inspired by classic music from old westerns.

# **5. Additional Features (Time Permitting)**

Ideal Game Session: There will be multiple levels per stage (if we decide to do more than one stage) that will slowly raise in difficulty, i.e. more/faster enemies. The player will begin with three lives and be able to gain lives whether through scoring a certain amount of points or if we can implement an enemy drop function that will drop a power-up. Types of power-ups could be but also not limited to a bomb that kills all enemies on the screen, an item that makes the player invincible for a short time, or an extra life.

Maps: By making maps that have obstacles such as bushes, rocks, and cacti, we can make the game more fun and interesting, along with complicating things for the player so they have to change their strategies each map change.

Sprite Graphics: By adding more sprite states for each object to show directionality and movement, we can make the graphics more fluid for a better player experience. This would involve synchronizing the inputs and movements of the sprites to their respective bit-map.

Enemy Movement: Once we have the general movement towards the player, we can experiment with other movements to see if any work better for the game. One idea is a percentage system where the enemy moves towards the player 75% of the time, but diverts either right or left 25% of the time. This could give the snakes more of a slither motion.

Death Animation: We could add in death animations for the snakes and the players.

More Enemy Types: It would be nice to have a variety of enemies that have varied durability and movement.

Difficulty: Another feature could be to increase the speed over all for difficulty or perhaps more difficult enemies spawn in.

Power Ups: As enemies die, some may drop power ups. These can be added in one at a time, possibly starting with the simple extra life drop in the form of a tiny stetson hat.