**Pseudocode**

Importing *pygame* and everything from *pygame* locals.

Set window width to one-thousand, window height to six-hundred, text color to black, frames per second to ~~sixty~~ **one-hundred-and-twenty**, missile size range from ten to forty, and missile speed range from one to four. **Level is set to one.**

terminate():

Quit pygame and exit through system method.

watiForPlayerToPressKey():

Begin a while true loop.

For an event going on in pygame…

if the event corresponds to quitting…

terminate the program.

If the event corresponds to pressing a key down…

If said key is Esc…

Terminate the program.

Elif said key is space bar…

Exit the function and progress in-game.

playerHasHitMissile(snailRect, missiles):

For all missiles…

If speed snail’s rectangle collides with a missile’s rectangle…

Set function to true.

Set function to false.

playerHasHitSpeedPowerUp(snailRect, speed\_power\_ups):

If speed snail’s rectangle collides with a speed power-up’s rectangle…

Set function to true.

Set function to false.

playerHasHitPointsPowerUp(snailRect, points-power-ups):

If speed snail’s rectangle collides with a points power-up’s rectangle…

Set function to true.

Set function to false.

drawText(text, font, surface, x, y):

Set text object to a rendered font of the text color and text.

Retrieve the text object’s rectangular properties.

Set the top-left of the rectangle to (x,y).

Blit the text onto the screen.

Set the snail’s image to the file provided (speed\_snail.png).

Retrieve the snail’s rectangular properties.

Set the missile’s image to the file provided (salt.png).

Set the speed power-up’s image to the file provided (speed\_power\_up\_orb.png)

Retrieve the speed power-up’s rectangular properties.

Set the points power-up’s image to the file provided (points\_power\_up\_orb.png)

Retrieve the points power-up’s rectangular properties.

Begin pygame initiation.

Start the clock.

Establish the surface to the co-ordinates defined above.

Set the window caption to Speed Snail.

**Set the mouse cursor to** ~~not~~ **appear.**

Initialize the power-up collection sound with its proper file (orb\_collect.wav)

Initialize the missile collision sound with its proper file (missile\_collision.wav)

Initialize the background music with its proper file (background\_music.mp3)

Set the font to the default system font at 48.

Set bg to the primary background (background.png)

Set begin\_bg to the beginning screen background (beginning\_screen.png)

Set start\_bg to the starting screen (starting\_screen.png)

Set instruct\_bg to the instructions screen (instructions.png)

Draw the beginning screen.

waitForPlayerToPressKey().

Draw the instructions screen.

waitForPlayerToPressKey().

Draw the starting screen.

waitForPlayerToPressKey().

Set hi\_score to zero.

Begin a while True loop.

Set facing to right.

Initialize the missile list.

**If the level is level one…**

Blit the first background.

**If the level is level two…**

**Blit the second background.**

**If the level is one…**

Set the score to zero.

**If the level is two…**

**Set the score to ten thousand.**

**Set the score multiplier to one.**

Set the snail speed to two.

Set the value of point orbs to five-hundred.

Set the missile spawning rate to fifty.

Set death to False.

**Set endless mode to False.**

Spawn the snail at (four-hundred, four-hundred).

Spawn the speed power orb in a random spot on screen.

Spawn the points power orb in a random spot on screen.

Set all movement values to False.

Set the missile add counter to zero.

Initialize the background music.

Begin a while True loop.

Constantly add the value of score multiplier to score.

For an event going on in pygames…

If the event corresponds to quitting…

terminate()

If the event corresponds to a key being pressed down…

If the key is the left arrow key or A…

Set facing to left.

Set the snail image to a mirrored version of itself.

Set moveRight to False and moveLeft to True.

If the key is the right arrow key or D…

Set facing to right.

Set the snail image to itself.

Set moveR*ig*ht to True and moveLeft to False.

If the key is the up arrow key or W…

If facing is equal to right...

Set snail image to more thrust.

Else:

Set snail image to more thrust and mirror.

Set moveUp to True and moveDown to False.

If the key is the down arrow key or S…

If facing is equal to right...

Set snail image to less thrust.

Else:

Set snail image to less thrust and mirror.

Set moveUp to False and moveDown to False.

If the event corresponds to a key being lifted…

If said key is Esc…

terminate()

If said key is the left arrow key or A…

Set the snail image to mirror itself.

Set moveLeft to False.

If said key is the right arrow key or D…

Set the snail image to itself.

Set moveRight to False.

If said key is the up arrow key or W…

Set moveUp to False.

If said key is the down arrow key or S…

Set moveDown to False.

If all movement is False…

If the facing is right…

Set the snail image to itself.

Else:  
 Set the snail image to mirror itself.

~~Set snail position to mouse cursor.~~

Add one to missile add counter.

If the missile add counter equals the missile spawning rate…

Set missile add counter to zero.

Set the missile size to a random value within the size range initialized above.

Set a new missile in a random vertical position on the screen on the far left side.

If moveLeft is True and snail’s rectangle’s left edge is greater than zero…

Jump the snail’s position a speed size pixel to the left.

If moveRight is True and snail’s rectangle’s right edge is less than the window width…

Jump the snail’s position a speed size pixel to the right.

If moveUp is True and snail’s rectangle’s top edge is greater than zero…

Jump the snail’s position a speed size pixel upwards.

If moveDown is True and snail’s rectangle’s bottom edge is less than four-hundred-and-ninety…

Jump the snail’s position a speed size pixel to the left.

For each missile…

Move the missile at the selected speed to the right.

For each missile…

If the missile goes past the right edge of the screen…

Delete the missile entity.

**If the level is level one…**

Draw the first background.

**Elif the level is level two…**

**Draw the second background.**

Draw the score value at the bottom of the screen.

Draw the high score value at the bottom of the screen, right of score.

~~Set the cursor position to the snail’s position.~~

If endless mode is on…

Draw the text saying endless mode is on.

Blit the snail on the surface.

Blit the speed power-up on the surface.

Blit the points power-up on the surface.

For each missile…

Blit the missile on the surface.

Update display.

If playerHasHitMissile() returns True…

If score’s value is greater than that of high score…

Set high score to score.

Play the collision sound.

Set Death to True.

Break the while True loop.

If playerHasHitSpeedPowerUp() returns True…

Play the orb collection sound.

Add one to the snail’s speed value.

Add one to the score multiplier.

Spawn in a new speed orb at a random location on-screen.

If playerHasHitSpeedPowerUp() returns True…

Play the orb collection sound.

Add the point orb value to the score.

Add five-hundred to the current point orb value.

Spawn in a new points orb at a random location on-screen.

**If the level is level one…**

If the score is greater than nine-thousand-and-ninety-nine…

Break the while True loop.

**Else:  
 If the score is greater than twenty-four-thousand-nine-hundred-and-ninety-nine…**

**Set endless mode to True.**

Set Main Clock to frames per second.

If death is True…

Display the text “You died! Press SPACEBAR to try again.” In the center of the screen.

**If the level is level two…**

**Set the score to ten-thousand.**

Update the display.

waitForPlayerToPressKey()

**Elif level is equal to one…**

**Display the text “Entering level two… Press SPACEBAR to continue.” In the center of the screen.**

**Set the background to the level two background image.**

**Set the missile image to that of the image provided (flame.png).**

**Set level to two.**

**Update the display.**

**waitForPlayerToPressKey()**