Natalie Garza

Fall 2018

SRS

# ☠🐛 Creepy Crawlers 🐛☠

tABLE OF cONTENTS

☠🐛 Creepy Crawlers 🐛☠ 1

✨Story: 3

✨Details/How to Play: 3

✨Components: 4

✨Functionality Divided: 7

✨ Level Plans: 8

✨ Code Plan and Design: 10

## ✨Story:

The idea driving the game is the fact that food-stressed caterpillars will resort to cannibalism. We can clearly make decisions and the idea itself will drive the game play. For example, a couple of rules that have stemmed from this idea:

* The bugs spawned will be sparser as the game progresses, thus making the players crawler win the game by eating other crawlers.
* Other crawlers will already be food-stressed and will automatically resort to eating the player.
* The longer crawlers will eat the smaller ones.
* You only have one life, in the caterpillar world it's do or die.

In the end the crawler must eat its fellow caterpillars to win the game, this is the only way if it wants to survive long enough to become a butterfly.

## ✨Details/How to Play:

* User steers crawler through a maze using the up, down, left, and right arrows on a keyboard.
* Crawlers can eat little bugs to grow their tail by one unit. Bugs are generated randomly on the edge of the map.
* If one crawler is bigger than another by at least one unit they may eat that one’s head and grow their own tail by one unit.
* Crawlers may nibble the tails of others to shorten their length. If the tail is nibbled towards the middle of their body, their entire tail from that point of contact is also removed.
* You win the game by reaching a tail size of 9, once you achieve this you are taken to the next level.
* User progresses through three different levels in the game. The higher levels will dictate the spawning frequencies of the bugs and the crawler types spawned.
* There are three varieties of crawlers that will differ in speed and smarts. Easy, medium, and hard.
* Depending on how far along the game the crawlers will spawn at longer lengths.
* Opponents don’t eat each other, they may only eat other bugs to grow their tail.
* If you eat a bug the new unit will be the same color as the head of the crawler, if you eat other crawlers head then the new unit will be the color of the head that was eaten.

**✨AI Aspects:**

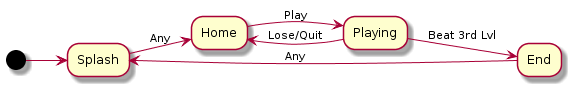
* Opponents measure the priority of eating bugs to increase their size VS eating the head of your crawler VS nibbling at your tail.
* Opponents hide or flee when you’re bigger than them or may decide to go after your tail if their decision-making algorithm thinks that they can get away with it safely.
* Bug spawning and crawler spawning is run by an algorithm and may be more sparse or rigorous depending on the level and length of player tail.
* Path finding algorithm towards player or bug, or when bugs lifecycle ends and they need to get to the nearest spawning point to die

## ✨Components:

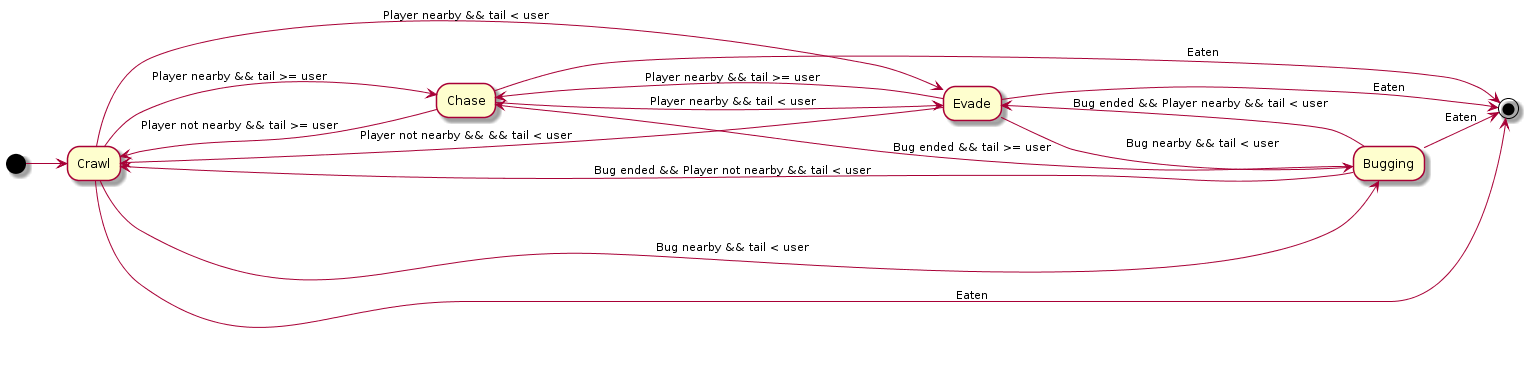
* Maze:
  + Consists of passages where crawlers will able to maneuver through. There will be one tunnel that will transport you to the other side of the map.
  + There will be tiles on the edge of the map where the bug prey will spawn from, the crawlers cannot go through these tiles.
  + Maps/maze should be Pac-man style where its symmetric down the middle.
  + Crawler passages will only be wide enough for one crawler to be able to pass at a time, opponent crawlers visually overlap each other in the passage if two are there at a time.
  + Users crawler bumping into a wall will stop crawlers’ motion, it will not move unless the user changes its direction. Crawlers in constant motion so if there is no obstacle in user’s way it will continue to crawl even when user is not pressing any keys.
* Crawlers:
  + There will be three varieties of opponent crawlers. Easy, Medium, and Hard they’ll be denoted by an “E”, “M”, and “H”.
  + Want to make the different behaviors of crawlers like Pac-man where the objective of each crawler changes depending on rank. H is like Blinky where its objective is to chase the player, M is like Pinky where its objective is to chase the players location 4 tiles ahead, and E is like Inky where its movement is unpredictable (either like Blinky, Pinky, or Clyde).
  + The different crawlers will have different behaviors depending on their rank. Speed and the algorithm defining their actions will vary.
  + Max length of opponent crawler is 12 units.
  + A crunch noise is made whenever a crawler body or head is munched on.
  + When a head is eaten the whole body of the crawler will also disappear instantly.
  + The user’s crawler is to trigger bug and crawler spawning based on the tail unit length. For example, user tail is at 9 units (nearly beating the level) this will trigger opponents of length 10 to appear, also their rank would be a rank higher than usual, bug spawning will be sparse, and crawler spawning will become dense.
  + Opponents do not eat each other.
* Bugs:
  + Spawn at the edge of the map.
  + Spawning frequency depends on the users tail length.
  + Each Bug has a lifecycle, they can freely roam around and if they aren’t eaten by the time their timer is up they must make their way to the closest spawning spot and disappear.
  + Lifecycles vary on the level and the length of the user’s tail.
  + A crunch noise is also made whenever the bugs are eaten.

**✨Finite State Machines:**

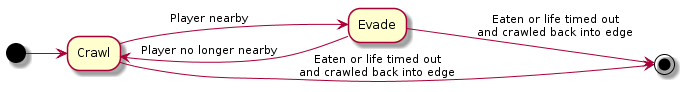
* Screens/Game:
  + Splash: I want to display the quote “Food-stressed caterpillars will resort to cannibalism”, vector graphic animation is displayed
    - Press any button to take you to home screen
  + Home: Play game instructions and ‘press z to start’ dialogue and an icon that can denote sound on/sound off. Sound toggled by pressing x.
    - Press z to take you to Play screen
  + Play: Map takes up whole screen and there will be a sound and z to quit dialogue.
    - Quit action takes you to Home screen
  + End: This is shown only when you beat the game. Maybe make it your crawler turning into a butterfly…
    - Press any button to take you to Splash screen



* Crawlers:
  + Crawl: Wander aimlessly.
    - Chase: Player nearby && tail >= user
    - Evade: Player nearby && tail < user
    - Bugging: Bug nearby && tail < user
    - Termination: Eaten
  + Chase: In pursuit of User. Transition to:
    - Evade: Player nearby && tail < user
    - Crawl: Player not nearby && tail >= user
    - Termination: Eaten
  + Bugging: In pursuit of Bug. Transition to:
    - Evade: Bug ended && Player nearby && tail < user
    - Chase: Bug ended && tail >= user
    - Crawl: Bug ended && Player not nearby && tail < user
    - Termination: Eaten
  + Evade: Running away or hiding from user. Transition to:
    - Bugging: Bug nearby && tail < user
    - Chase: Player nearby && tail >= user
    - Crawl: Player not nearby && && tail < user
    - Termination: Eaten



* Bugs:
  + Crawl: Wander aimlessly. Transition to:
    - Evade: Player nearby
    - Termination: Eaten or life timed out and crawled back into edge
  + Evade: Running away or hiding from predators.
    - Crawl: Player no longer nearby
    - Termination: Eaten or life timed out and crawled back into edge



## ✨Functionality Divided:

1. Map
   1. Split up into the necessary screens
   2. Wrap around effect, Pac-man style
   3. Walls that you can’t pass through
2. Player
   1. Move around map using up, down, left, right buttons
   2. Movement rules applied ex. Always moving unless run into wall, slow down the longer the tail
   3. Grow and shrink accordingly
3. Crawlers
   1. Move around map in coherence with AI and FSM
   2. Spawning manager
   3. Easy, medium, and hard types
   4. Grow and shrink accordingly
4. Bugs
   1. Move around map in coherence with AI and FSM
   2. Spawning manager
5. Levels
   1. Layout the three different levels
   2. Pre-program triggers for enemies/bugs and spawning amount of each type of enemy
   3. Losing mechanism
6. Screens
   1. FSM for different screens and game states implemented ex. MenuStateInit
   2. Buttons or hotkeys to move through screens work ex. Sound, play, try again, and quit
7. Art
   1. Choose color palette and style
   2. Player, enemies, bug, map, screens, game title, buttons, etc. Drawn out in full detail whereas before it was ugly and very simple
   3. Motion sprites
   4. Vector graphics for splash screen?
8. Sound
   1. Game song, different for each level
   2. Winning song
   3. Other sounds ex. Collision, eat, lose, win, change direction...
9. Effects
   1. Screen fade in/out
   2. Blood squirts
   3. Camera shaking
   4. Final butterfly sequence
   5. Between level animated scenes

## ✨ Level Plans:

When a player dies, they are directed to the home screen, player has to go through all three levels without dying to win the game and see the final butterfly sequence.

There will be an animated sequence between levels. So, two total sequences. Perhaps it could be the crawler building their chrysalis and some dialogue.

4 bug spawning spots and 2 crawler spawning spots

**\*\* Tables below are to give general idea of implementation, these values can/should be tweaked \*\***

* + Changes between levels:
    - Spawning frequencies; increase crawler and decrease bug
    - Type of crawler spawned
    - Music, intensity increases as you go through levels
  + Changes within a level, depends on player length:
    - Length of crawlers spawned
    - Spawning frequencies; increase crawler and decrease bug
    - Type of crawlers spawned
    - Music speed
  + Level Specifications: Concepts & rough estimates
    - Level 1:
      * E --> M crawler enemies spawned \*\*
      * Enemy crawler length spawned same length +0 --> +1 \*\*
      * Max enemy length in level = 10 = (9+1)
      * Max enemies at any given time = 2

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Player  Length | Crawlers Spawned | Crawler  Freq (secs) | Crawler  Length  (units) | Prob of E type | Prob of M type | Prob of H type | Bugs Spawned | Bug Freq (secs) | Bug Lifecycle (secs) |
| 1 (head) | 1 | 10 | 1 | 1.00 | 0 | 0 | 1 | 10 | 20 |
| 2 | 2 | 10 | 2 | .90 | .10 | 0 | 1 | 15 | 20 |
| 3 | 2 | 10 | 3 | .80 | .20 | 0 | 1 | 20 | 20 |
| 4 | 3 | 10 | 4 | .70 | .30 | 0 | 1 | 25 | 20 |
| 5 | 3 | 10 | 5 | .60 | .40 | 0 | 1 | 30 | 15 |
| 6 | 3 | 10 | 7 | .50 | .50 | 0 | 1 | 35 | 10 |
| 7 | 4 | 10 | 8 | .20 | .80 | 0 | 0 | 0 | 0 |
| 8 | 4 | 5 | 9 | .10 | .90 | 0 | 0 | 0 | 0 |
| 9 | 4 | 5 | 10 | 0 | 1.00 | 0 | 0 | 0 | 0 |

* + - Level 2:
      * E --> M --> H OR E--> M, H \*\*
      * Spawn +1 --> +2 \*\*
      * Max enemy length = 11 = (9+2)
      * Max enemies at any given time = 2

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Player  Length | Crawlers Spawned | Crawler  Freq (secs) | Crawler  Length  (units) | Prob of E type | Prob of M type | Prob of H type | Bugs Spawned | Bug Freq (secs) | Bug Lifecycle (secs |
| 1 (head) | 2 | 10 | 2 | .90 | .10 | 0 | 1 | 20 | 20 |
| 2 | 3 | 10 | 3 | .70 | .30 | 0 | 1 | 30 | 20 |
| 3 | 3 | 10 | 4 | .50 | .50 | 0 | 1 | 40 | 15 |
| 4 | 4 | 10 | 5 | .30 | .70 | 0 | 1 | 50 | 10 |
| 5 | 4 | 10 | 7 | .10 | .90 | 0 | 1 | 60 | 10 |
| 6 | 4 | 5 | 8 | 0 | .90 | .10 | 1 | 90 | 10 |
| 7 | 5 | 5 | 9 | 0 | .70 | .30 | 0 | 0 | 0 |
| 8 | 5 | 5 | 10 | 0 | .50 | .50 | 0 | 0 | 0 |
| 9 | 5 | 5 | 11 | 0 | .30 | .70 | 0 | 0 | 0 |

* + - Level 3:
      * M --> H \*\*
      * Spawn +2 --> +3 \*\*
      * Max enemy length = 12 (9+3)
      * Max enemies at any given time = 3

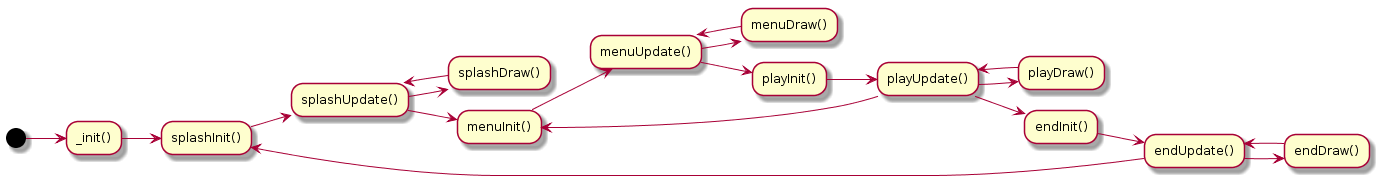
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Player  Length | Crawlers Spawned | Crawler  Freq (secs) | Crawler  Length  (units) | Prob of E type | Prob of M type | Prob of H type | Bugs Spawned | Bug Freq (secs) | Bug Lifecycle (secs) |
| 1 (head) | 3 | 10 | 2 | .30 | .70 | 0 | 1 | 20 | 15 |
| 2 | 3 | 10 | 3 | .20 | .80 | 0 | 1 | 40 | 15 |
| 3 | 4 | 10 | 5 | .10 | .70 | .20 | 1 | 60 | 10 |
| 4 | 4 | 10 | 6 | 0 | .50 | .50 | 1 | 90 | 10 |
| 5 | 4 | 10 | 7 | 0 | .30 | .70 | 1 | 120 | 10 |
| 6 | 5 | 5 | 8 | 0 | .10 | .90 | 0 | 0 | 0 |
| 7 | 5 | 5 | 10 | 0 | 0 | 1.00 | 0 | 0 | 0 |
| 8 | 5 | 5 | 11 | 0 | 0 | 1.00 | 0 | 0 | 0 |
| 9 | 5 | 5 | 12 | 0 | 0 | 1.00 | 0 | 0 | 0 |

\* Spawning spots randomly chosen

\*\* Proportional to player length

## ✨ Code Plan and Design:

To represent the four game states, there will be four different sets of Init, Update, and Draw functions. Like in a typical game loop all the initialization needed for a particular state will be done in the corresponding init function and the game will go between the draw and update function until some condition is met within update. Once that condition is met the game can continue to the next set of state functions.



* \_init(): This is the entry point to the game, to start the game FSM we want to make a call to splashInit()
* SplashInit(): initialization code for splash screen, most of it should come from the vector animation making tool (http://hotmessgames.com/poly/)
  + Hex(x): convert x to hex
* SplashUpdate(): Most of this code should come from the vector animation tool. This should also be watching for the user to press any key this will signal the transition to menuInit() which leads to the menu state of the game.
* SplashDraw(): Draw the vector animation as well as any sprites/text you want to incorporate into the splash screen.
* MenuInit(): Initialize code related to the menu state, such as menu text and sound toggling.
* MenuUpdate(): Update menu graphics.
  + UpdateSettings(): checks to see if the z or x buttons are pressed. If z is pressed go to play state, if x is pressed toggle sound.
* MenuDraw(): Draw menu graphics, also if sound is toggled change icon appropriately.
* PlayInit(): Initialize all game play code, such as crawler & player tables.
  + MakeGame(): keeps track of game info such as win(bool), lastBugSpawn(time), level(int)...
  + MakePlayer(): tracks x,y location of player, speed, length...
  + MakeCrawlers(): tracks all the opponent crawlers in the game
  + MakeBugs(): tracks all the bugs in the game
  + MakeMap(): keeps info on map hard-coded data such as spawning areas
* PlayUpdate(): Update all the entities in the game
  + UpdateSettings(): checks to see if player wants to quit or toggle sound
  + UpdateGame()
  + UpdateCrawlers()
  + UpdateBugs()
  + UpdateMap()
* PlayDraw():
  + DrawCrawlers()
  + DrawBugs()
  + DrawMap()
* EndInit(): If player wins the third level then they are taken to this state. All the code necessary to run the vector animation tool will be here, similar to the splash screen code.
* EndUpdate(): Calculations for animation.
* EndDraw(): Draw animation

Pico-8 Map: Can use up to 16 screens of 128x128 bit cells without needing to use the shared memory space. Here’s a layout of the necessary screens:

1. Splash
2. Menu
3. Level 1
4. Linking sequence 1
5. Level 2
6. Linking sequence 2
7. Level 3
8. Butterfly ending sequence
9. Dying/Quitting animation?