Enemy Behaviors

Knox Game Design

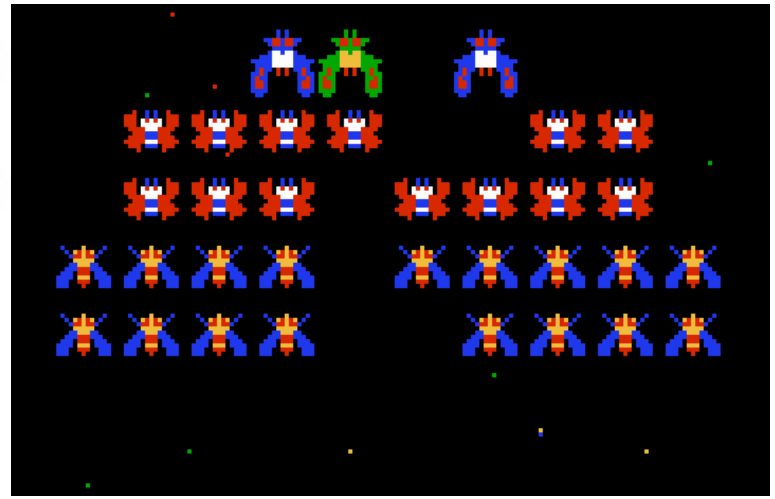
May 2022

Levi D. Smith

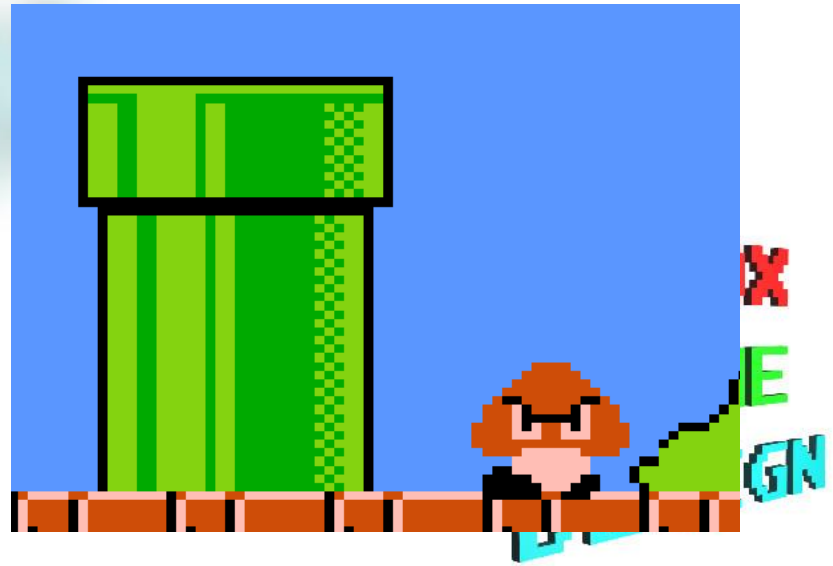
Basic Enemy Class

- Position (float): x, y
- Size (int): w, h
- Velocity (float): vel_x, vel_y
- Sprite images

- Move back and forth
  - x += vel_x * delta_time

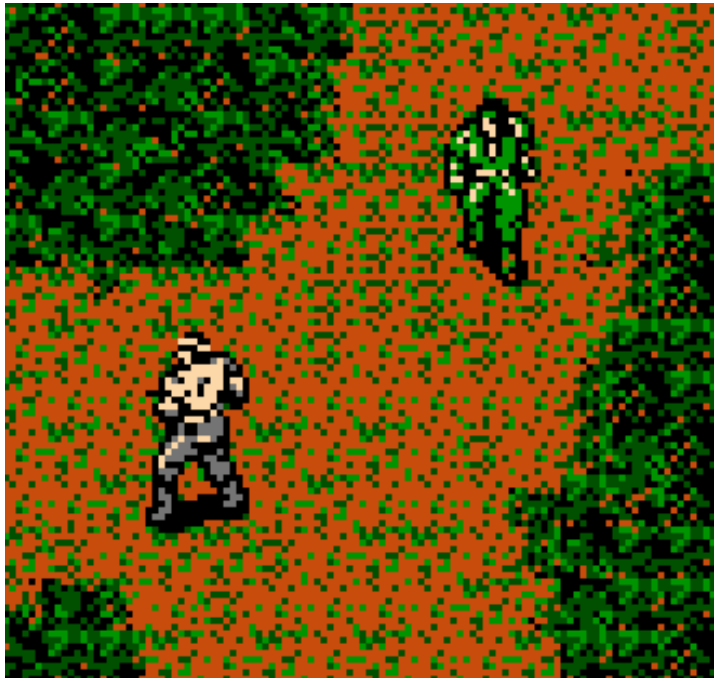


- Timer based
  - Requires countdown variables (float)
    - Current countdown
    - Max countdown
  - vel_change_countdown = max_countdown
  - vel_change_countdown -= delta_time
  - if (vel_change_countdown <= 0)
    - vel_x = vel_x * -1
    - vel_change_countdown += max_countdown

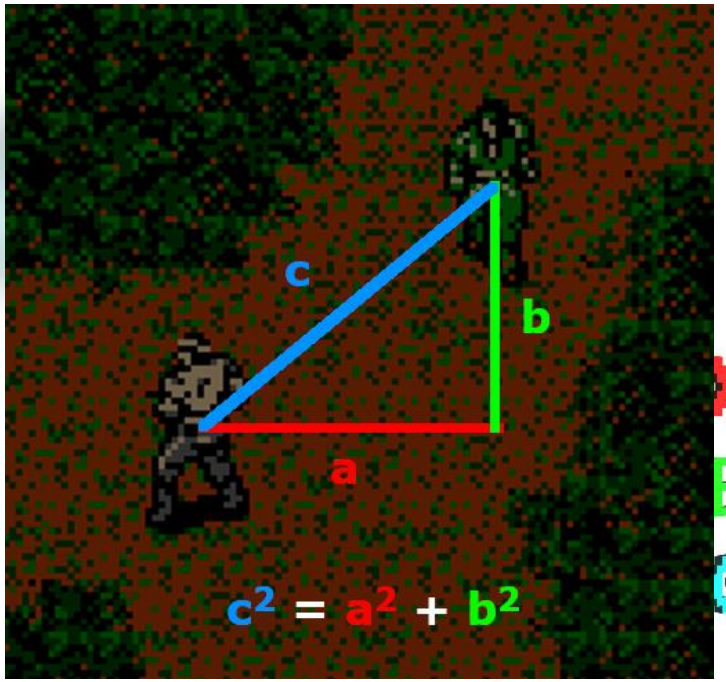


- Collision based
  - Requires collision detection method
  - Check collision with all solid objects
  - If collide with object
  - vel_x = vel_x * -1
  - Simplified version of "bounce"

- Proximity, Alert and chase
  - variables: alert distance (float), is chasing (bool)
  - if distance(player.pos, enemy.pos) < alert_distance
    - enemy.isChasing = true
  - if (enemy.isChasing)
    - enemy.pos += move_towards(player.pos) * delta_time

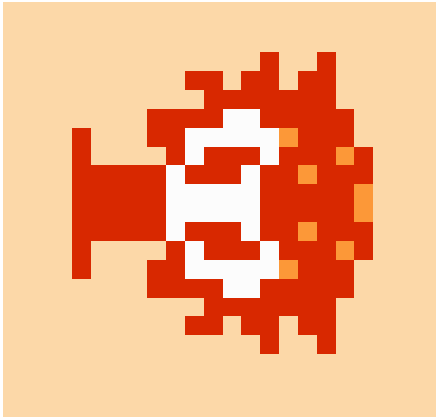


- distance = $((x_1 - x_2)^2 + (y_1 - y_2)^2)^{1/2}$
  - *(Pythagroean theorem)*



- Chase end conditions
  - Distance from player
    - constant: chase end distance (float)
    - if (distance(player.pos, enemy.pos) > stop_chasing_distance)
      - enemy.isChasing = false
  - Timer based
    - variable: chase end countdown (float), constant: max chase end countdown(float)
    - if (chase_end_countdown <= 0f)

- Move, Stop, Change Direction
- Variables: move countdown, max move countdown, stop countdown, max stop countdown
- def change_direction()
  - vel_x = speed * cos(rand_angle)
  - Vel_y = speed * cos(rand_angle)
  - random angle from 0 to 360 degrees, set x_vel and y_vel using speed * sine/cosine(angle)



- if (wait_countdown > 0)
  - wait_countdown -= delta_time
  - if (wait_countdown <= 0)
    - change_direction()
    - moving_countdown = max_moving_countdown

- if (moving_countdown > 0)
  - x += vel_x
  - y += vel_y
  - moving_countdown -= delta_time
  - if(moving_countdown <= 0)
    - wait_countdown = max_wait_countdown

- Enemy Children - orbital
  - Variables: lifetime, list of children, orbit radius, orbit speed
  - x: orbit radius * cosine((orbit_speed * lifetime) + angle_offset)
  - y:orbit radius * sine((orbit_speed * lifetime) + angle_offset)
    - angle_offset = (child_number / child_count) * 2 * PI
- Can stack behaviors
  - Orbital children + move back and forth
  - Orbital children + move/stop/change direction