Project 4:
Pacman Dodge

Subject : Introduction to Visual Media Programming
Professor : Yongduek, Seo
Major : Art&Technology
Student No. : 20181227
Name : Hojin, Kim
Submission Date : 08 Jan 2023

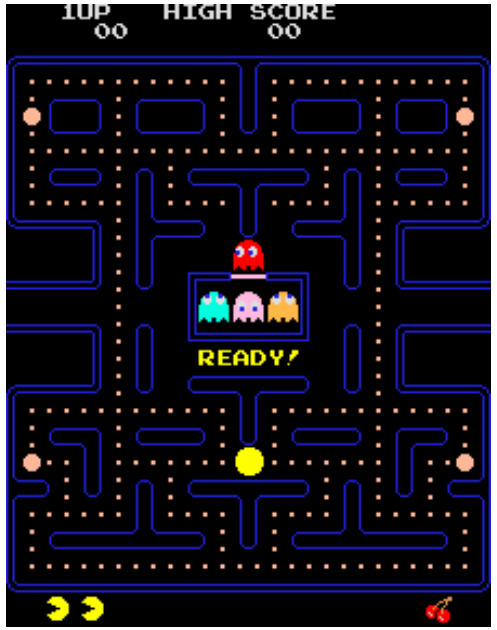
Contents

1. Project Summary

Pacman Dodge

After having several experiences on creating games and other applications with pygame, I thought it would be possible to create my own game by utilizing what I had learned until now. As I had interest in arcade games, and as I wanted to make a unique game rule, I decided to create a game which is a combination of two different game: Pacman and Dodge.

The pacman will move around the window and gather dots while avoiding crowds of ghosts. After the pacman gains a power-up item, ghosts will become vulnerable and run away from the pacman. When pacman hits a ghost without having any power-up, the pacman will die and the game will be over.



2. Game Rule (Changes)

Control

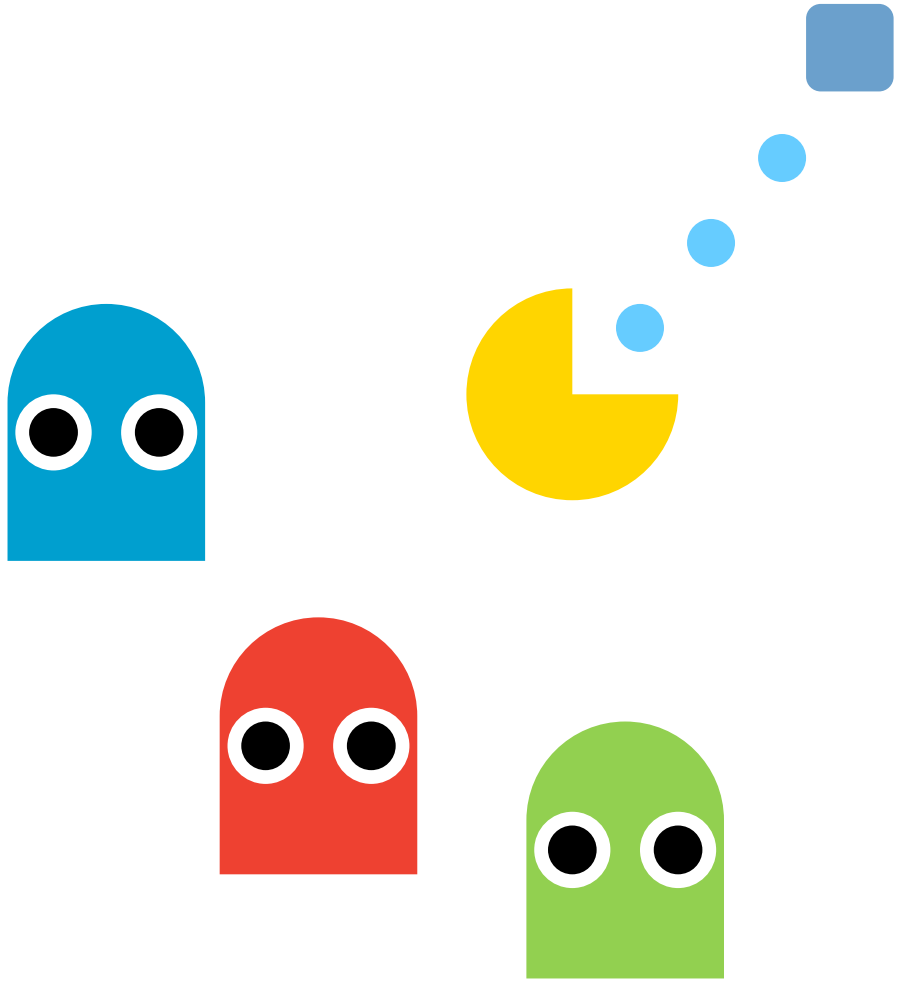
- Mouse ~~click~~ movement – move pacman to mouse location

Goal

- Collect as many dots as possible
- Avoid any contact with crowd of ghosts
- Defeat ghosts after receiving power-up

Gameover Condition

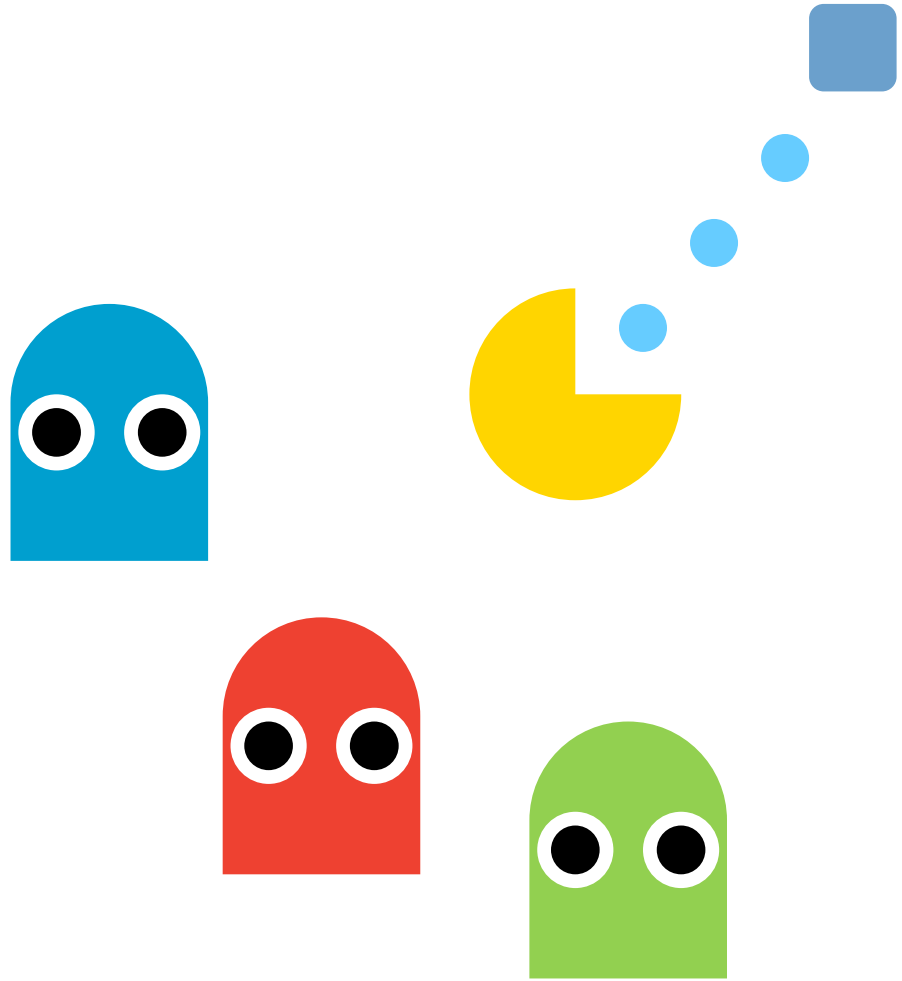
- When pacman hits any ghost without having a power-up
- ~~When pacman does not collect any dots for certain amount of time~~



2. Game Rule (Changes)

Ghost

- When pacman does not have any power-up, ghosts will follow pacman.
- Pacman having a contact with a ghost on this state will lead to the gameover screen.
- When pacman has a power-up, ghosts will run away from pacman.
- Pacman having a contact with a ghost on this state will lead to gain additional point and ~~destroy the ghost.~~ make it run away to it’s respawn point.
- The number of spawned ghosts will increase when the game continues.



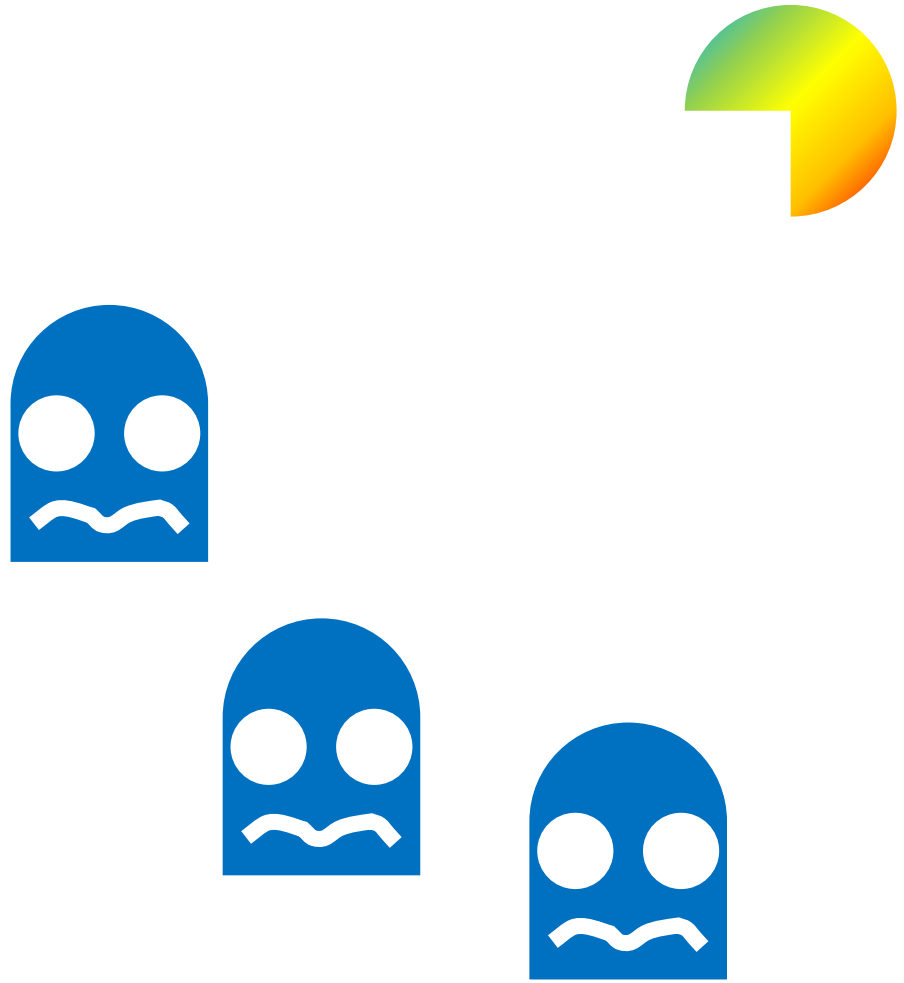
3. Game Rule (New)

Score

- Collecting a dot will give pacman 1 point.
- Defeating a ghost while having a power-up will give pacman 3 points.

Power-up

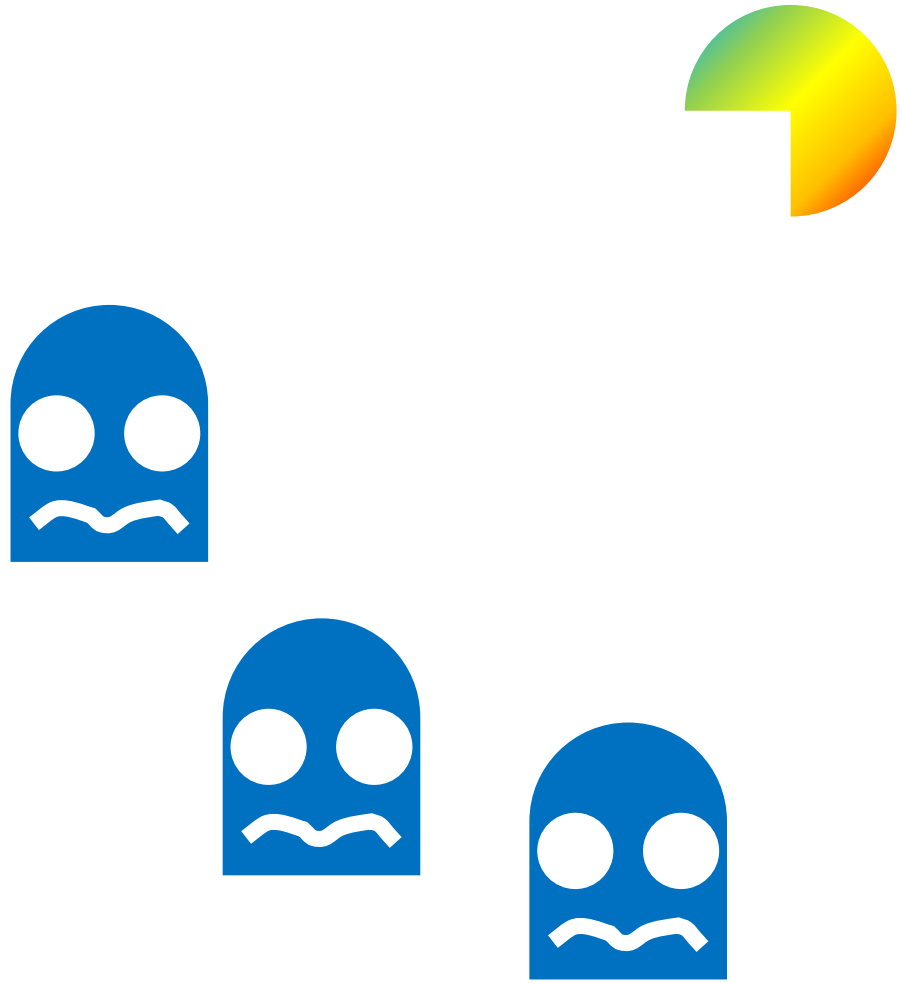
- To spawn a power-up, certain number of dots is needed.
- Only after reaching to the target number of dots, power-up will spawn on the field.



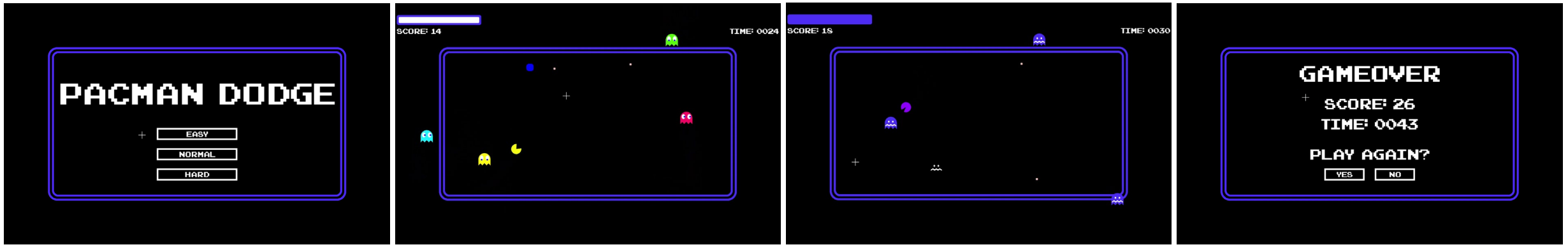
3. Game Rule (New)

Getting Harder Over Time

- Gaining a power-up will temporarily make pacman become stronger and will increase the number of dots on the field by 1.
- However, after power-up ends, the number of ghosts will increase by 1, and the number needed to spawn a power-up will increase dramatically.



4. Screenshot & Demo video



Github: https://github.com/Bbummo/Project4_20181227_HojinKim

Demo Video: https://youtu.be/UZ8G5IT1DPE

5. Process

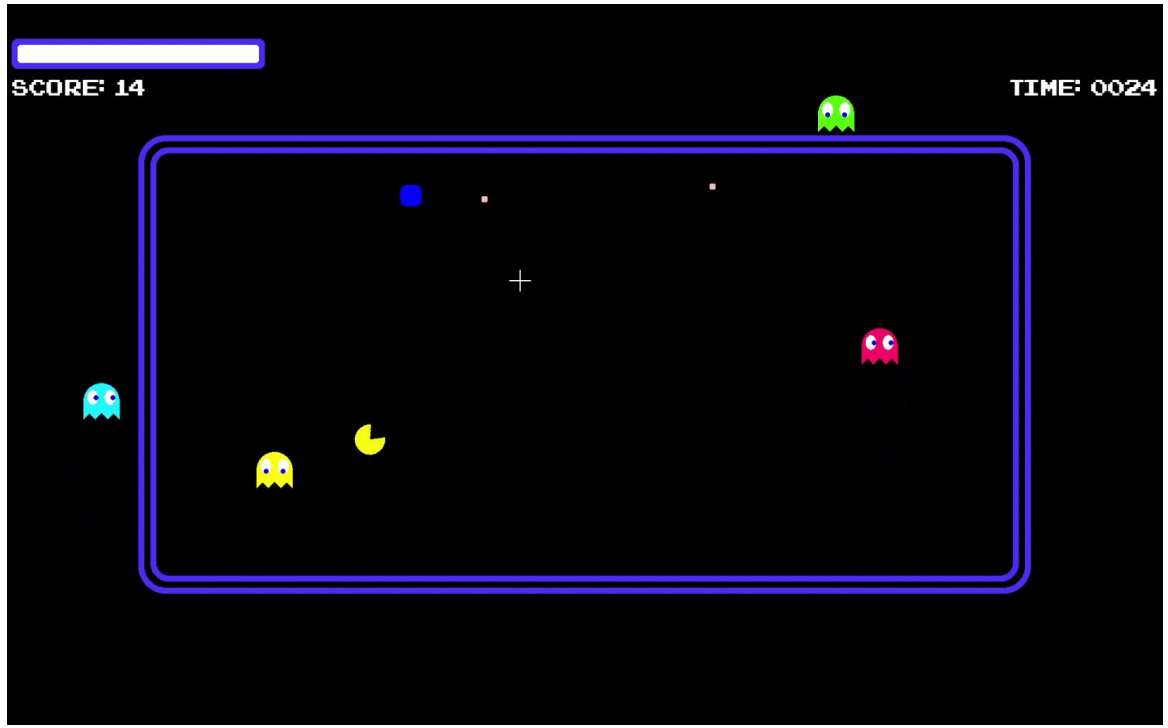
Main Lobby

- This is the main lobby of this game.
- User can control a pacman by moving his/her mouse around the screen.
- There are three kinds of difficulties in this game:
  - **Easy**: Starts with 2 dot requirement to spawn power-up, starts with 2 ghosts.
  - **Normal**: Starts with 3 dot requirement to spawn power-up, starts with 3 ghosts.
  - **Hard**: Starts with 4 dot requirement to spawn power-up, starts with 4 ghosts.
- Once a difficulty is selected by mouse click, the game will start.

5. Process

Ingame - 1

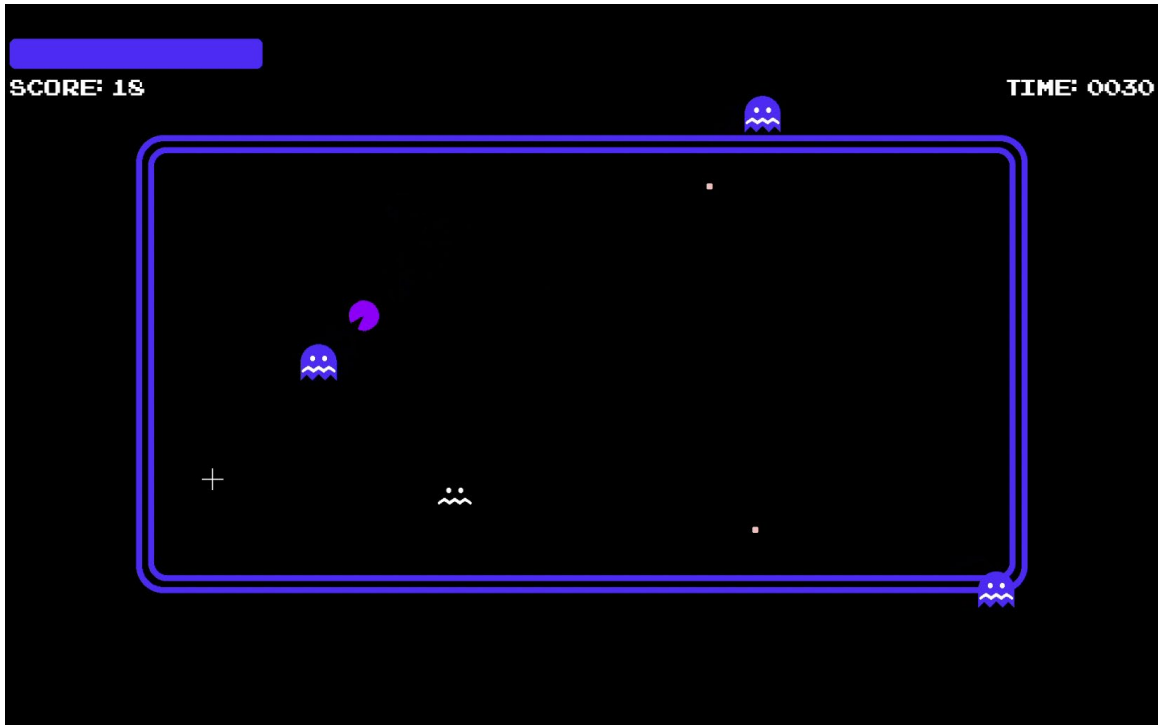
- This is the ingame screen of this game.
- User can control a pacman by moving his/her mouse around the screen.
- Ghosts will frequently follow pacman and try to catch it.
- User can collect dots on the screen.
- One collected dot gives +1 point.
- When dots are collected to certain amount and when it fills the power gauge, power-up item appears on the screen.



5. Process

Ingame - 2

- When the pacman eats a power-up, it becomes invincible and weakens ghosts.
- On this state, pacman can defeat ghosts by running into them.
- One defeated ghost gives +3 point.
- When a ghost is defeated, its body becomes transparent, and it runs away to its spawned location.
- After power-up effect is finished, the number of total ghost increases by 1, the total number of spawned dots on the field increases by 1, and the number of required dots to spawn power-up increases.



5. Process

Gameover

- When the pacman make a contact with a ghost without havin a power-up, the game ends.
- The gameover screen displays player’s score and time played.
- The gameover screen asks player if he/she would play the game again.
  - **Yes**: return to main lobby page
  - **No**: the program terminates

6. Technical Highlights

Class Collider_circle

- This is a simple circle collider which is made up with two factors: position and radius.
- This collider is used on checking any collision between two objects.

```
262  class Collider_circle:
263      def __init__(self, _pos, _radius):
264          self.pos = _pos
265          self.radius = _radius
266
267      def draw(self, _screen): # for debug
268          pygame.draw.circle(_screen, RED, self.pos, self.radius, 1)
```

6. Technical Highlights

Class Pacman

- This class creates a playable character, pacman.

- This class saves information such as:
  - Position
  - Movement Direction
  - Movement Speed
  - Is Powered-up
  - Current Animation Image

- This class contains function such as:
  - Rotation
  - Movement
  - Movement Animator
  - Draw

```python
class Pacman():
    def __init__(self,):
        self.pos = [WINDOW_WIDTH/2, WINDOW_HEIGHT/2]
        self.direction = 0
        self.speed = 5.0
        self.radius = 25
        stayInSafezone(self, SAFEZONE_PACMAN)
        self.collider = Collider_circle(self.pos, self.radius)
        self.isPowerUp = False
        self.imgArrIndex = 1
        self.imgArrIndexVel = 1
        self.imgArr = []
        for i in range(7):
            tempImg = pygame.image.load(os.path.join(assets_path, 'pacman_'+ str(i) +'.png'))
            self.imgArr.append(tempImg)

    def rotate(self, _mousePos):
        tempAngle = angleOf2Dot(self.pos, _mousePos)
        if tempAngle != -1:
            self.direction = tempAngle

    def move(self, _mousePos):
        rad = np.deg2rad(self.direction)
        distance = distanceOf2Dot(self.pos, _mousePos)
        moveAmount = self.speed
        if distance < self.speed:
            moveAmount = distance
        amountX = moveAmount * np.cos(rad)
        amountY = - moveAmount * np.sin(rad)
        self.pos = [self.pos[0]+amountX, self.pos[1]+amountY]
        stayInSafezone(self, SAFEZONE_PACMAN)
        self.collider.pos = self.pos
    def animator_move(self,): # Waka waka animation
        if self.imgArrIndex >= 6 or self.imgArrIndex <= 0:
            self.imgArrIndexVel *= -1 # Animate in reverse order
        self.imgArrIndex += self.imgArrIndexVel

    def update(self, _mousePos, _isPowerUp):
        self.isPowerUp = _isPowerUp
        self.rotate(_mousePos)
        self.move(_mousePos)

    def draw(self, _screen):
        self.animator_move()
        img_r = pygame.transform.rotate(self.imgArr[self.imgArrIndex], self.direction)
        if isPowerUp: # Strong state
            img = img_r.copy()
            tempHueVal = frameTime/60
            tempHueVal = tempHueVal - int(tempHueVal)
            tempColor = hsv2rgba(tempHueVal)
            reColor(img, (244,246,0,255), tempColor)
            _screen.blit(img, img.get_rect(center= self.pos))
        else: # Normal state
            img = img_r
        _screen.blit(img, img.get_rect(center= self.pos))
        #self.collider.draw(_screen)
```

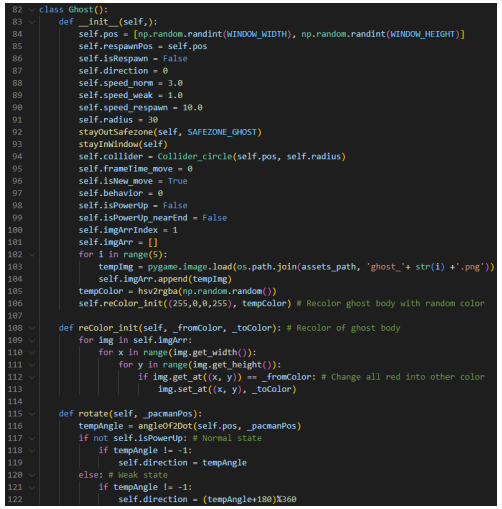
6. Technical Highlights

Class Ghost

- This class creates an enemy character, ghost.

- This class saves information such as:
  - Position
  - Respawn Position
  - Movement Behavior
  - Movement Direction
  - Movement Speed (Norm, Weak, Respawn)
  - Is Powered-up
  - Current Animation Image

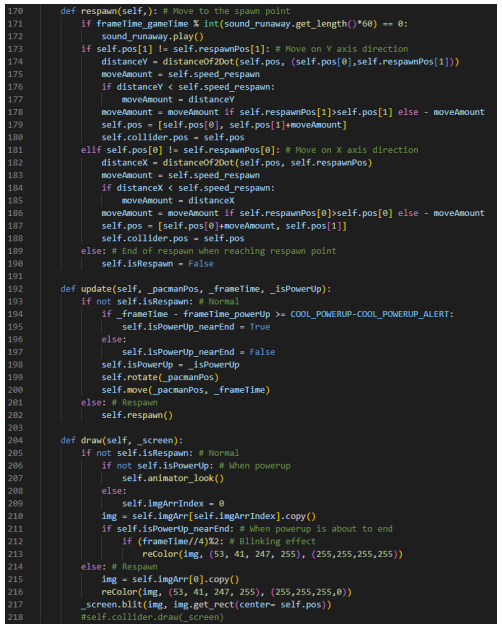
- This class contains function such as:
  - Rotation
  - Movement Behavior Selector
  - Movement
  - Respawn Movement
  - Movement Animator
  - Body Recolor Painter
  - Draw



```
class Ghost():
    def __init__(self,):
        self.pos = [np.random.randint(WINDOW_WIDTH), np.random.randint(WINDOW_HEIGHT)]
        self.respawnPos = self.pos
        self.isRespawn = False
        self.direction = 0
        self.speed_norm = 3.0
        self.speed_weak = 1.0
        self.speed_respawn = 10.0
        self.radius = 30
        stayOutSafezone(self, SAFEZONE_GHOST)
        stayInWindow(self)
        self.collider = Collider_circle(self.pos, self.radius)
        self.frameTime_move = 0
        self.isNew_move = True
        self.behavior = 0
        self.isPowerUp = False
        self.isPowerUp_nearEnd = False
        self.imgArrIndex = 1
        self.imgArr = []
        for i in range(5):
            tempImg = pygame.image.load(os.path.join(assets_path, 'ghost_'+ str(i) +'.png'))
            self.imgArr.append(tempImg)
        tempColor = hsv2rgba(np.random.random())
        self.reColor_init((255,0,0,255), tempColor) # Recolor ghost body with random color

    def reColor_init(self, _fromColor, _toColor): # Recolor of ghost body
        for img in self.imgArr:
            for x in range(img.get_width()):
                for y in range(img.get_height()):
                    if img.get_at((x, y)) == _fromColor: # Change all red into other color
                        img.set_at((x, y), _toColor)

    def rotate(self, _pacmanPos):
        tempAngle = angleOf2Dot(self.pos, _pacmanPos)
        if not self.isPowerUp: # Normal state
            if tempAngle != -1:
                self.direction = tempAngle
        else: # Weak state
            if tempAngle != -1:
                self.direction = (tempAngle+180)%360
```



```
    def respawn(self,): # Move to the spawn point
        if frameTime_gameTime % int(sound_runaway.get_length()*60) == 0:
            sound_runaway.play()
        if self.pos[1] != self.respawnPos[1]: # Move on Y axis direction
            distanceY = distanceOf2Dot(self.pos, (self.pos[0],self.respawnPos[1]))
            moveAmount = self.speed_respawn
            if distanceY < self.speed_respawn:
                moveAmount = distanceY
            moveAmount = moveAmount if self.respawnPos[1]>self.pos[1] else - moveAmount
            self.pos = [self.pos[0], self.pos[1]+moveAmount]
            self.collider.pos = self.pos
        elif self.pos[0] != self.respawnPos[0]: # Move on X axis direction
            distanceX = distanceOf2Dot(self.pos, self.respawnPos)
            moveAmount = self.speed_respawn
            if distanceX < self.speed_respawn:
                moveAmount = distanceX
            moveAmount = moveAmount if self.respawnPos[0]>self.pos[0] else - moveAmount
            self.pos = [self.pos[0]+moveAmount, self.pos[1]]
            self.collider.pos = self.pos
        else: # End of respawn when reaching respawn point
            self.isRespawn = False

    def update(self, _pacmanPos, _frameTime, _isPowerUp):
        if not self.isRespawn: # Normal
            if _frameTime - frameTime_powerUp >= COOL_POWERUP-COOL_POWERUP_ALERT:
                self.isPowerUp_nearEnd = True
            else:
                self.isPowerUp_nearEnd = False
            self.isPowerUp = _isPowerUp
            self.rotate(_pacmanPos)
            self.move(_pacmanPos, _frameTime)
        else: # Respawn
            self.respawn()

    def draw(self, _screen):
        if not self.isRespawn: # Normal
            if not self.isPowerUp: # When powerup
                self.animator_look()
            else:
                self.imgArrIndex = 0
            img = self.imgArr[self.imgArrIndex].copy()
            if self.isPowerUp_nearEnd: # When powerup is about to end
                if (frameTime//4)%2: # Blinking effect
                    reColor(img, (53, 41, 247, 255), (255,255,255,255))
        else: # Respawn
            img = self.imgArr[0].copy()
            reColor(img, (53, 41, 247, 255), (255,255,255,0))
        _screen.blit(img, img.get_rect(center= self.pos))
        #self.collider.draw(_screen)
```

```
    def move(self, _pacmanPos, _frameTime):
        if _frameTime - self.frameTime_move > 60: # Shift behavior every 1 second
            self.isNew_move = True
        if self.isNew_move: # Select new behavior
            self.frameTime_move = _frameTime
            self.moveBehaviorSelect() # 0-front, 1-left, 2-back, 3- right
            self.isNew_move = False
        self.direction += self.behavior*90
        rad = np.deg2rad(self.direction)
        distance = distanceOf2Dot(self.pos, _pacmanPos)
        speed = self.speed_norm if not self.isPowerUp else self.speed_weak
        if distance < speed:
            moveAmount = distance
        else:
            moveAmount = speed
        amountX = moveAmount * np.cos(rad)
        amountY = - moveAmount * np.sin(rad)
        self.pos = [self.pos[0]+amountX, self.pos[1]+amountY]
        stayInWindow(self)
        self.collider.pos = self.pos

    def animator_look(self,): # Image selection according to moving direction
        if self.direction>45 and self.direction<315:
            self.imgArrIndex = int((self.direction+45)//90 + 1)
        else:
            self.imgArrIndex = 1

    def moveBehaviorSelect(self,):
        # Behavior ratio
        FRONT = 0.50
        LEFT = 0.20
        BACK = 0.10
        #RIGHT = 0.20
        if not self.isPowerUp: # Normal state
            tempBehavior = np.random.random()
            if tempBehavior < FRONT: # front
                self.behavior = 0
            elif tempBehavior < FRONT+LEFT: # left
                self.behavior = 1
            elif tempBehavior < FRONT+LEFT+BACK: # back
                self.behavior = 2
            else: # right
                self.behavior = 3
        else: # Weak state
            self.behavior = 0 # only go front
```

6. Technical Highlights

Class Dot

- This class creates a score-related item, dot.
- When a pacman collides with this dot, this dot does not disappear; instead, it relocates itself somewhere in the screen.

```
220  class Dot():
221      def __init__(self,):
222          self.pos = [np.random.randint(WINDOW_WIDTH), np.random.randint(WINDOW_HEIGHT)]
223          self.radius = 5
224          stayInSafezone(self, SAFEZONE_DOT)
225          self.collider = Collider_circle(self.pos, self.radius)
226          self.img = pygame.image.load(os.path.join(assets_path, 'dot.png'))
227
228      def relocate(self):
229          self.pos = [np.random.randint(WINDOW_WIDTH), np.random.randint(WINDOW_HEIGHT)]
230          stayInSafezone(self, SAFEZONE_DOT)
231          self.collider.pos = self.pos
232
233      def draw(self, _screen):
234          _screen.blit(self.img, self.img.get_rect(center= self.pos))
235          #self.collider.draw(_screen)
```

6. Technical Highlights

Class Power

- This class creates a power-up item, power.
- When a pacman collides with this power, this power disappear(actually, it relocates itself out of the screen.).
- When certain condition is met, it relocates itself somewhere in the screen.

```
237 class Power():
238     def __init__(self,):
239         self.pos = [-100, -100]
240         self.radius = 20
241         self.collider = Collider_circle(self.pos, self.radius)
242         self.img = pygame.image.load(os.path.join(assets_path, 'power.png'))
243 
244     def relocate(self): # Let power displayed inside the screen
245         self.pos = [np.random.randint(WINDOW_WIDTH), np.random.randint(WINDOW_HEIGHT)]
246         stayInSafezone(self, SAFEZONE_DOT)
247         self.collider.pos = self.pos
248 
249     def exclude(self): # Let power displayed out of screen
250         self.pos = [-100, -100]
251         self.collider.pos = self.pos
252 
253     def draw(self, _screen):
254         img = self.img.copy()
255         tempHueVal = frameTime/60
256         tempHueVal = tempHueVal - int(tempHueVal)
257         tempColor = hsv2rgba(tempHueVal)
258         reColor(img, (255,0,0,255), tempColor)
259         _screen.blit(img, img.get_rect(center= self.pos))
260         #self.collider.draw(_screen)
```

6. Technical Highlights

Angle/Distance between Two Dots

- Function angleOf2Dots() calculates an angle between two dots.
  - It receives two coordinate arrays and returns degree angle from 0 to 359.9.

- Function distanceOf2Dots() calculates a distance between two dots.
  - It receives two coordinate arrays and returns a distance between these two dots.

```
322  def angleOf2Dot(_start, _end): # Angle between two dots in degree
323      differenceX = _end[0] - _start[0]
324      differenceY = _end[1] - _start[1]
325      if differenceX != 0:
326          if differenceX > 0 and differenceY <= 0: # 0~90
327              angle = -np.rad2deg(np.arctan(differenceY/differenceX))
328          elif differenceX < 0 and differenceY < 0: # 90~180
329              angle = 180 - np.rad2deg(np.arctan(differenceY/differenceX))
330          elif differenceX < 0 and differenceY >= 0: # 180~270
331              angle = 180 - np.rad2deg(np.arctan(differenceY/differenceX))
332          elif differenceX > 0 and differenceY > 0: # 270~360
333              angle = 360 - np.rad2deg(np.arctan(differenceY/differenceX))
334      else:
335          if differenceY > 0:
336              angle = 270.0
337          elif differenceY < 0:
338              angle = 90.0
339          else: # When _end point is same as _start point
340              angle = -1
341      return angle
342
343  def distanceOf2Dot(_start, _end): # Distance between two dots
344      differenceX = _end[0] - _start[0]
345      differenceY = _end[1] - _start[1]
346      return np.power(np.power(differenceX,2) + np.power(differenceY,2), 0.5)
```

6. Technical Highlights

Animator

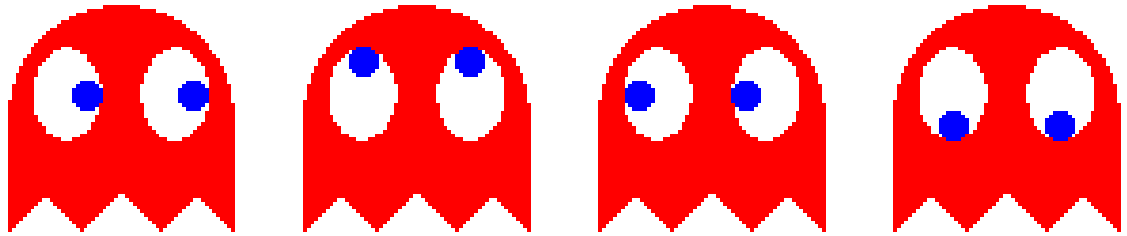
- In order to make dynamic movement of characters, frame-by-frame animation is applied on this project.

- **animator_move()**
  - Applied on Pacman class
  - Continuously dlsplays next frame image repetitively

- **animator_look()**
  - Applied on Ghost class
  - Shifts image according to its moving direction


```
57      def animator_move(self,): # Waka waka animation
58          if self.imgArrIndex >= 6 or self.imgArrIndex <= 0:
59              self.imgArrIndexVel *= -1 # Animate in reverse order
60          self.imgArrIndex += self.imgArrIndexVel
```

```
145     def animator_look(self,): # Image selection according to moving direction
146         if self.direction>45 and self.direction<315:
147             self.imgArrIndex = int((self.direction+45)//90 + 1)
148         else:
149             self.imgArrIndex = 1
```

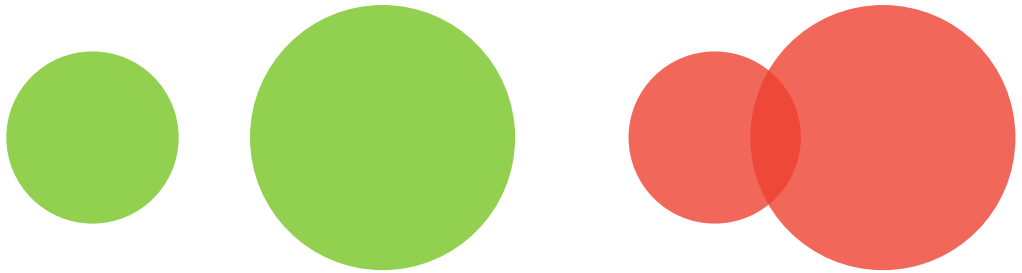


6. Technical Highlights

Collision Check between Two Objects

- Collision between two objects is needed to be recognized in order to initiate certain event or assign special effect to an object.
- is2ObjCollide() function receives two class objects with circle colliders and returns True/False depending on if the collision actualy happened.
- The function detects collision when the distance between two circle collider is smaller than the sum of two colliders' radius.

```
348 def is2ObjCollide(_obj1, _obj2): # Is two circle collider collides
349     obj1_pos = _obj1.collider.pos
350     obj1_radius = _obj1.collider.radius
351     obj2_pos = _obj2.collider.pos
352     obj2_radius = _obj2.collider.radius
353     obj_distance = distanceOf2Dot(obj1_pos, obj2_pos)
354     if obj_distance < obj1_radius +obj2_radius:
355         return True
356     else:
357         return False
```



6. Technical Highlights

Position to Inside/Outside of Range

- It could be a complete mayhem if any object can spawn on any empty place of the screen.

- stayInSafezone(), stayOutSafezone(),and stayInWindow(), restricts the position of certain object into some designated area.
  - **stayInSafezone()**: restricts position to stay only inside the safe zone.
  - **stayOutSafezone()**: restricts position to stay only outsde the safe zone.
  - **stayInWindow()**: restricts position to stay only inside the window screen.

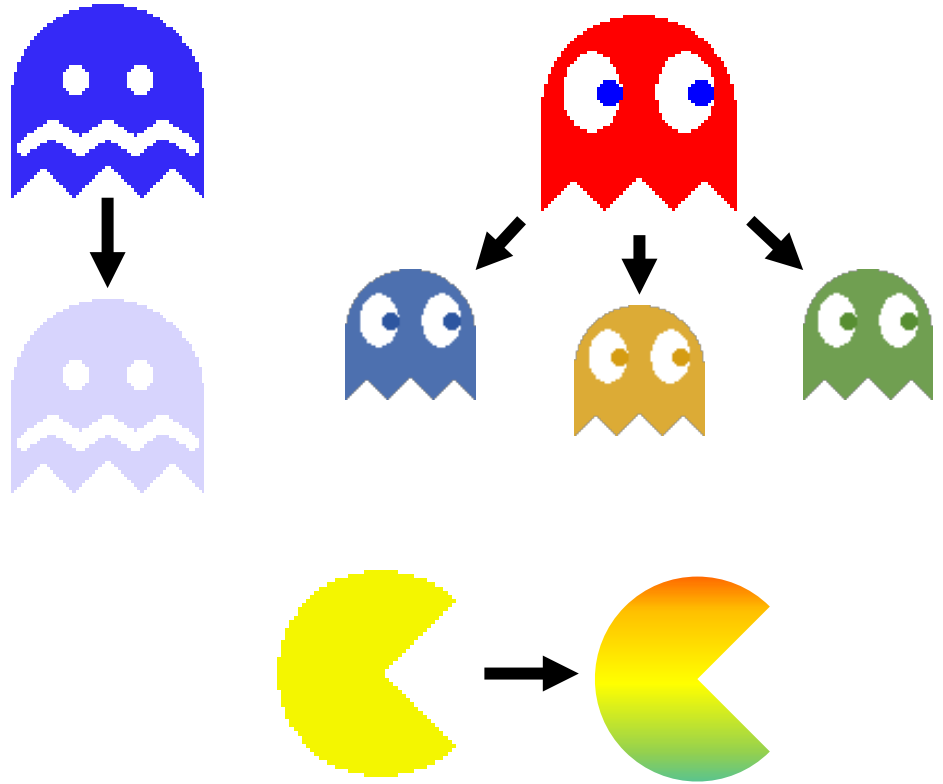
```
359  def stayInSafezone(_obj, _safeZone): # Change position to inside of safe zone
360      obj_pos = _obj.pos
361      obj_radius = _obj.radius
362      if obj_pos[0] < _safeZone+obj_radius:
363          _obj.pos[0] = _safeZone+obj_radius
364      elif obj_pos[0] > WINDOW_WIDTH-_safeZone-obj_radius:
365          _obj.pos[0] = WINDOW_WIDTH-_safeZone-obj_radius
366      if obj_pos[1] < _safeZone+obj_radius:
367          _obj.pos[1] = _safeZone+obj_radius
368      elif obj_pos[1] > WINDOW_HEIGHT-_safeZone-obj_radius:
369          _obj.pos[1] = WINDOW_HEIGHT-_safeZone-obj_radius
370
371  def stayOutSafezone(_obj, _safeZone): # Change position to outside of safe zone
372      obj_pos = _obj.pos
373      obj_radius = _obj.radius
374      if obj_pos[0] < WINDOW_WIDTH/2:
375          if obj_pos[0] > _safeZone-obj_radius:
376              _obj.pos[0] = _safeZone-obj_radius
377      else:
378          if obj_pos[0] < WINDOW_WIDTH-_safeZone+obj_radius:
379              _obj.pos[0] = WINDOW_WIDTH-_safeZone+obj_radius
380      if obj_pos[1] < WINDOW_HEIGHT/2:
381          if obj_pos[1] > _safeZone-obj_radius:
382              _obj.pos[1] = _safeZone-obj_radius
383      else:
384          if obj_pos[1] < WINDOW_HEIGHT-_safeZone+obj_radius:
385              _obj.pos[1] = WINDOW_HEIGHT-_safeZone+obj_radius
386
387  def stayInWindow(_obj,): # Change position to inside of window screen
388      obj_pos = _obj.pos
389      obj_radius = _obj.radius
390      if obj_pos[0] < obj_radius:
391          _obj.pos[0] = obj_radius
392      elif obj_pos[0] > WINDOW_WIDTH-obj_radius:
393          _obj.pos[0] = WINDOW_WIDTH-obj_radius
394      if obj_pos[1] < obj_radius:
395          _obj.pos[1] = obj_radius
396      elif obj_pos[1] > WINDOW_HEIGHT-obj_radius:
397          _obj.pos[1] = WINDOW_HEIGHT-obj_radius
```

6. Technical Highlights

Swap Image's Pixel Color

- Along with animation, special color shift is needed to make game look more diverse and create special effects.
- reColor() function reads a given image and swaps every certain color into a given color.
- reColor() can change a color into a solid color, or it can also change the transparency of the color, too.

```
406  def reColor(_img, _fromColor, _toColor): # Swap certain pixel into other color
407      for x in range(_img.get_width()):
408          for y in range(_img.get_height()):
409              #print(_img.get_at((x, y)))
410              if _img.get_at((x, y)) == _fromColor:
411                  _img.set_at((x, y), _toColor)
```



The End