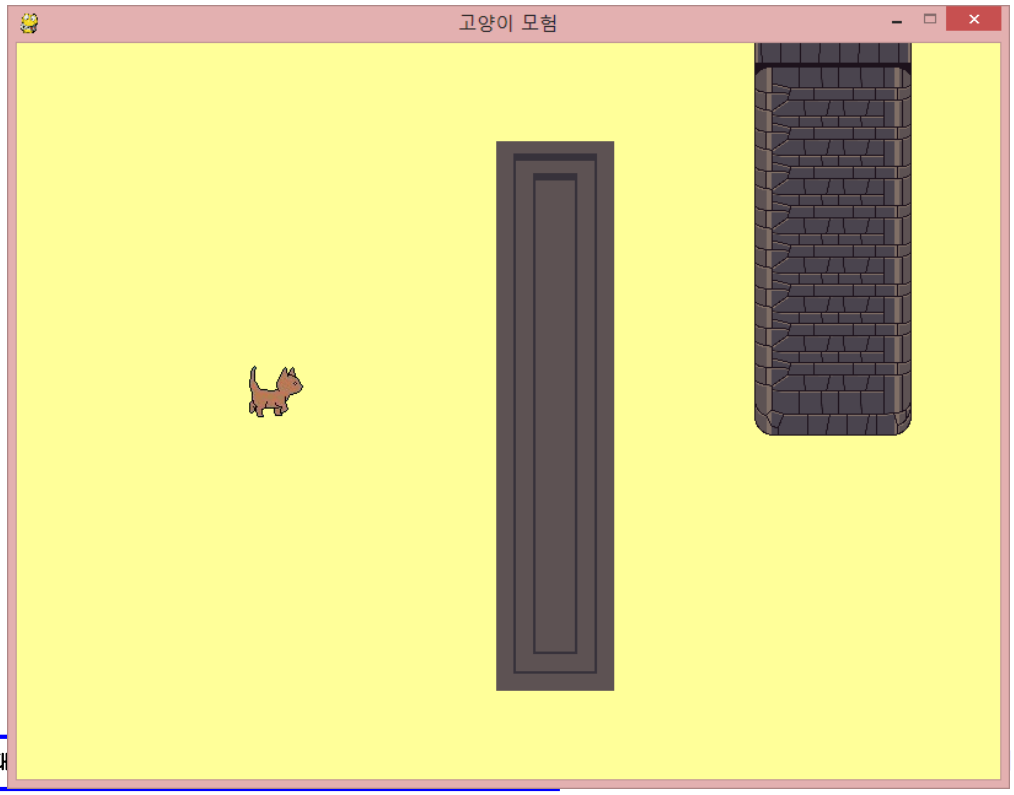
게임 예: 고양이 모험

순천향대학교 컴퓨터공학과
이 상 정

학습 내용

- 고양이가 여러 배경을 통과하는 어드벤처 게임 유형
- 객체 지향 프로그래밍
  - 장애물 정의 부모 클래스: GraphicWall
    - 자식 클래스: StoneWall, StoneWall2, Tree1, Tree2, WaterWall, GardenWall, TallGrass, FallGrass
  - 플레이어 고양이 정의 클래스: Player
- 스프라이트 활용
- 타일로 구성된 지도 이미지 파일에서 그림 추출하여 배경 장애물 표시
  - 11장 비트맵 그래픽스의 타일 이미지 추출 참조
  - 3개의 배경 화면
- 화살표 키 누르면 해당 방향으로 고양이 이동
- 장애물 충돌 시 이동 멈춤

배경 장면 1



배경 장면 2

배경 장면3

장애물 정의 부모 클래스 – GraphicWall

- 장애물 정의 부모 클래스, GraphicWall
  - 타일로 구성된 지도 이미지 파일(terrain_atlas.png)에서 그림 추출하여 배경 장애물 표시
  - 스프라이트 클래스로 부터 상속
  - 개별 장애물 클래스의 부모 클래스
  - setGraphic() 메소드
    - 이미지를 추출 하여 여러 개 복사하여 그리는 함수
      - 지정된 크기가 타일 크기의 배수인 경우 여러 개 복사
    - 타일 위치/크기, 스크린 위치/크기 인수로 전달
      def setGraphic(self,tilex,tiley,tilewidth,tileheight,x,y,width,height):
  - setGraphic2() 메소드
    - 중간 부분이 확장된 타일 이미지 추출,
    - 타일 위치, 스크린 위치/크기 인수로 전달
      def setGraphic2(self,tilex,tiley,x,y,width,height):
    - 11장 비트맵 그래픽스의 타일 이미지 추출 참조

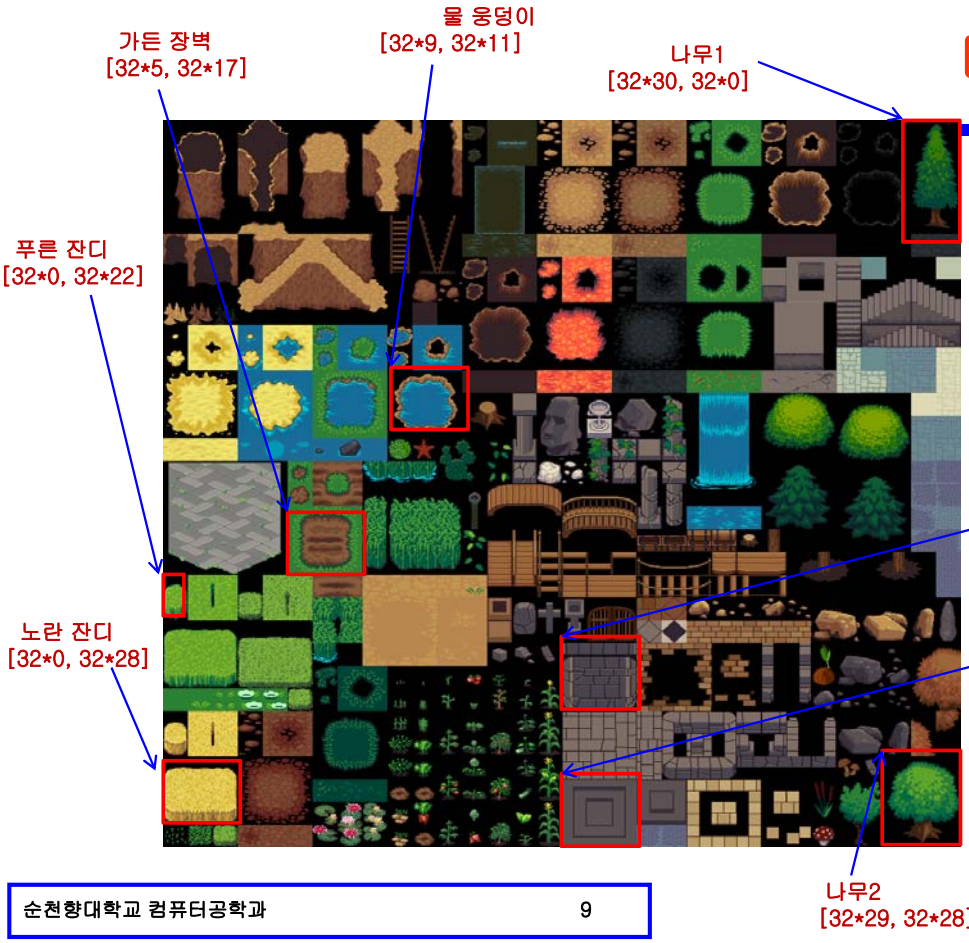
GraphicWall 클래스

```
# 장애물을 정의하는 클래스
# Sprite 부모 클래스를 상속
class GraphicWall(pygame.sprite.Sprite):
    # 이미지 추출 여러 개 복사 함수, 타일 위치/크기, 스크린 위치/크기 인수로 전달
    def setGraphic(self,tilex,tiley,tilewidth,tileheight,x,y,width,height):
        # 이미지 적재
        myimage = pygame.image.load("terrain_atlas.png").convert()
        # 스크린에 표시될 이미지 Surface 생성
        self.image = pygame.Surface([width, height])

        # 타일 크기 배수로 복사
        for row in range(height//tileheight):
            for column in range(width//tilewidth):
                self.image.blit(myimage,[column*tilewidth,row*tileheight],[tilex,tiley,tilewidth,tileheight])

        # 이미지 크기의 rect 객체 지정
        self.rect = self.image.get_rect()
        # 장애물의 스크린 위치 지정
        self.rect.x = x
        self.rect.y = y
        # 투명키 설정
        self.image.set_colorkey(black)
```

험

```
# 중간 부분이 확장된 타일 이미지 추출, 타일 위치 스크린 위치/크기 인수로 전달
    def setGraphic2(self,tilex,tiley,x,y,width,height):
        .........
        .........
        ## 1행 타일
        # 1행 1열 타일
        self.image.blit(myimage,[0,0],[tilex,tiley,32,32])
        # 1행 중간열 타일
        for column in range(width//32-2):
            self.image.blit(myimage,[(column+1)*32,0],[tilex+32,tiley,32,32])
        # 1행 마지막열 타일
        self.image.blit(myimage,[(width//32-1)*32,0],[tilex+64,tiley,32,32])

        ## 중간행 타일
        for row in range(height//32-2):
            self.image.blit(myimage,[0,(row+1)*32],[tilex,tiley+32,32,32])
            # row 행의 중간열 타일
            for column in range(width//32-2):
                self.image.blit(myimage,[(column+1)*32,(row+1)*32],[tilex+32,tiley+32,32,32])
            # row 행의 마지막 열 타일
            self.image.blit(myimage,[(width//32-1)*32,(row+1)*32],[tilex+64,tiley+32,32,32])

        ## 마지막행 타일
        # 마지막행 1열 타일
        self.image.blit(myimage,[0,(height//32-1)*32],[tilex,tiley+64,32,32])
        # 마지막행 중간열 타일
        for column in range(width//32-2):
            self.image.blit(myimage,[(column+1)*32,(height//32-1)*32],[tilex+32,tiley+64,32,32])
        # 마지막행 마지막열 타일
        self.image.blit(myimage,[(width//32-1)*32,(height//32-1)*32],[tilex+64,tiley+64,32,32])
         .........
```

타일 이미지

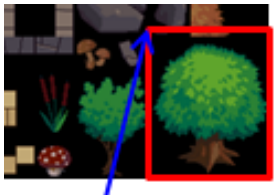


- 타일
  32 픽셀 x 32 픽셀
- 이미지 파일
  1024 픽셀 x 1024 픽셀
  32 타일 x 32 타일
  (32 타일 = 32 x 32 픽셀
  = 1024 픽셀)

게임 예: 고양이 모험

setGraphic() 메소드 사용 예

```
# 나무2 클래스
# GraphicWall 부모 클래스를 상속
class Tree2(GraphicWall):
    # 생성자, 스크린 위치 인수로 전달
    def __init__(self,x,y):
        # 부모 클래스 생성자 호출
        pygame.sprite.Sprite.__init__(self)
        # 스크린 이미지 크기
        width=96          # 3 타일
        height=128*2          # 4 타일 * 2, 2배 높이 지정
        # 타일 이미지 크기
        tilewidth=96
        tileheight=128
        # 타일 이미지 위치
        tilex=32*29
        tiley=32*28
        # 이미지 추출
        self.setGraphic(tilex,tiley,tilewidth,tileheight,x,y,width,height)
```

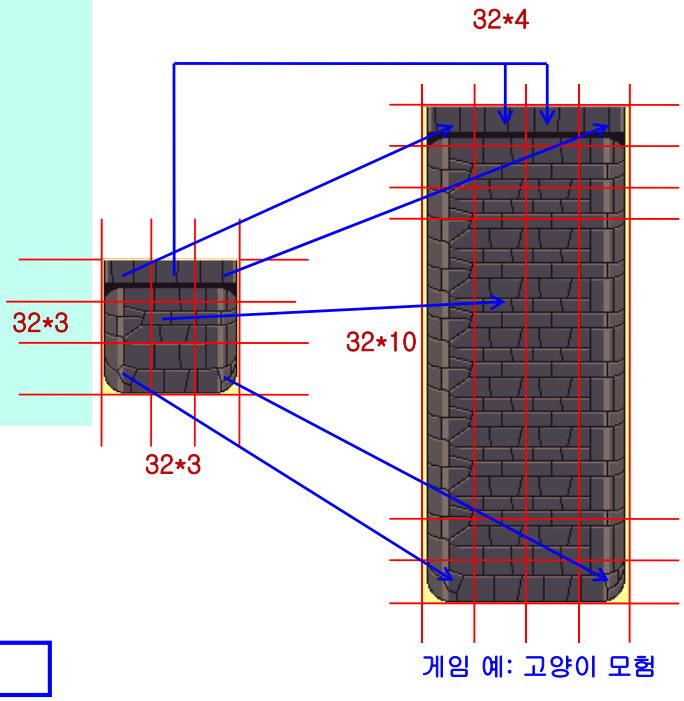


나무2
[32*29, 32*28]

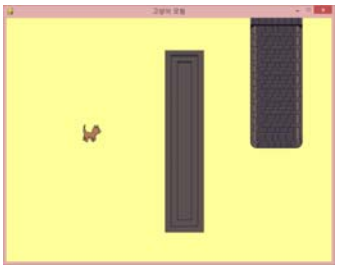
게임 예: 고양이 모험

setGraphic2() 메소드 사용 예

```
# 벽돌 장애물 클래스
# GraphicWall 부모 클래스를 상속
class StoneWall2(GraphicWall):
    # 생성자, 스크린 위치/크기 인수로 전달
    def __init__(self,x,y,width,height):
        # 부모 클래스 생성자 호출
        pygame.sprite.Sprite.__init__(self)
        # 타일 이미지 위치
        tilex=32*16
        tiley=32*23
        # 중간 확장 이미지 추출
        self.setGraphic2(tilex,tiley,x,y,width,height)
```



게임 예: 고양이 모험



배경 장면 정의 함수

```
# 배경 장면1
def setupRoomOne():
    # 스프라이트 그룹 생성
    wall_list=pygame.sprite.Group()

    # 석조 장애물 추가
    wall=StoneWall(390,80, 96, 448)
    wall_list.add(wall)
    # 벽돌 장애물
    wall=StoneWall2(600,0, 128, 320)
    wall_list.add(wall)

    # 스프라이트 그룹 리턴
    return wall_list
```

```
# 배경 장면2
def setupRoomTwo():
    # 스프라이트 그룹 생성
    wall_list=pygame.sprite.Group()

    # 나무1 추가
    wall=Tree1(100,100)
    wall_list.add(wall)
    # 나무2 추가
    wall=Tree2(300,250)
    wall_list.add(wall)
    # 잔디 장벽 추가
    wall=TallGrass(100,400,64,160)
    wall_list.add(wall)
    # 나무1 추가
    wall=Tree1(500,160)
    wall_list.add(wall)
    # 노란 잔디 장벽 추가
    wall=FallGrass(600,128,192,160)
    wall_list.add(wall)
    # 나무1 추가
    wall=Tree1(700,350)
    wall_list.add(wall)

    # 스프라이트 그룹 리턴
    return wall_list
```

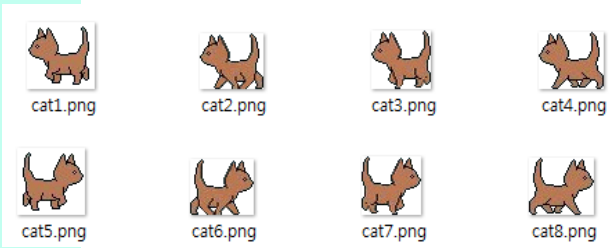
```
# 배경 장면3
def setupRoomThree():
    # 스프라이트 그룹 생성
    wall_list=pygame.sprite.Group()

    # 물 웅덩이 추가
    wall=WaterWall(64,64,192,192)
    wall_list.add(wall)
    # 가든 장벽 추가
    wall=GardenWall(128,325,256,192)
    wall_list.add(wall)
    # 나무1 추가
    wall=Tree1(520,256)
    wall_list.add(wall)

    # 스프라이트 그룹 리턴
    return wall_list
```

플레이어 고양이 정의 클래스 – Player

- 플레이어 고양이 정의 클래스, Player
  - 고양이 이미지 적재, 이동속도 벡터 값 설정, 위치 및 이미지 갱신
  - 스프라이트 클래스로 부터 상속
  - 생성자, __init__()
    - 8개의 고양이 이미지를 리스트에 적재
  - changespeed() 메소드
    - 이동속도 벡터 값 change_x, change_y 값 설정
    - 고양이 이동방향(오른쪽, 왼쪽, 위, 아래) 및 속도 표시
  - update() 메소드
    - 충돌 감지
    - 매 4 프레임 간격으로 고양이 8개 이미지 중 하나를 선택

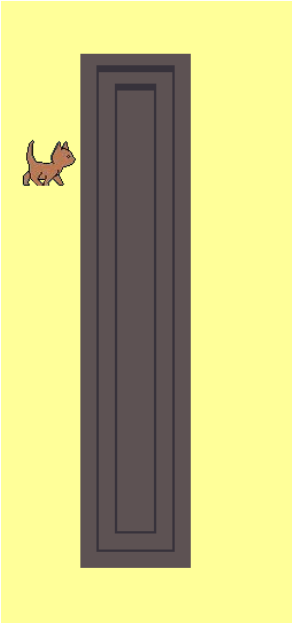
Player 클래스 – 이미지 적재

```
# 플레이어 고양이를 정의하는 클래스
# Sprite 부모 클래스를 상속
class Player(pygame.sprite.Sprite):
    # 클래스 속성
    # 고양이 이동속도 벡터 값 초기화, 왼쪽(음수) 오른쪽(양수) 방향
    change_x=0
    change_y=0
    # 고양이 이미지 선택을 위한 프레임 카운터, 4개 프레임 마다 이미지 선택
    frame = 0

    # 생성자
    def __init__(self):
        # 부모 클래스 생성자 호출
        pygame.sprite.Sprite.__init__(self)

        # 고양이 이미지 리스트 초기화
        self.images=[]
        # cat1.png ~ cat8.png 고양이 이미지 적재
        for i in range(1,9):
            img = pygame.image.load("cat"+str(i)+".png").convert()
            img.set_colorkey(white)
            self.images.append(img)

        # 디폴트로 고양이 images[0] 사용
        self.image = self.images[0]
        # 이미지 크기의 rect 객체 지정
        self.rect = self.image.get_rect()
```



Player 클래스 – 이동속도 및 충돌 감지

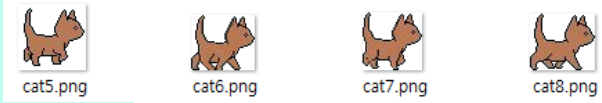
```
# 이동속도 벡터 값 설정
  def changespeed(self,x,y):
      self.change_x += x
      self.change_y += y

  # 고양이 위치와 이미지 갱신
  def update(self, walls):
      # x 축 새 위치 설정, 충돌 시 복구를 위해  이전 위치 저장
      old_x = self.rect.x
      new_x = old_x+self.change_x
      self.rect.x = new_x
      # x 방향 이동 시 충돌 여부 감지
      collide = pygame.sprite.spritecollide(self, walls, False)
      if collide:
          # 충돌인 경우 이 전 위치 복구
          self.rect.x = old_x

      # y 축 새 위치 설정, 충돌 시 복구를 위해  이전 위치 저장
      old_y = self.rect.y
      new_y = old_y+self.change_y
      self.rect.y = new_y
      # y 방향 이동 시 충돌 여부 감지
      collide = pygame.sprite.spritecollide(self, walls, False)
      if collide:
          # 충돌인 경우 이전 위치 복구
          self.rect.y=old_y
```



게임 예: 고양이 모험

Player 클래스 – 고양이 이미지 선택

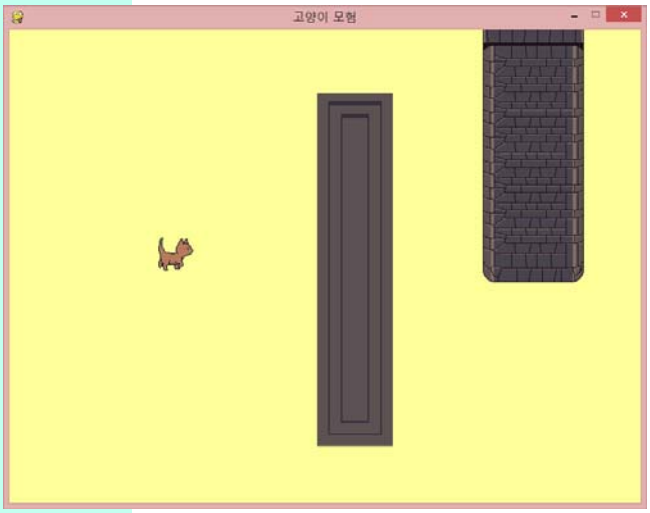
```
      # 왼쪽 방향 이동 중인 경우
      if self.change_x < 0:
          # 프레임 카운터 증가
          self.frame += 1

          # 4 프레임 간격으로 고양이 이미지(images[0]~images[3]) 변경
          # image[3] 이 후의 경우(프레임이 12(3*4)보다 크면) 프레임 리셋
          if self.frame > 3*4:
              self.frame = 0

          # 매 4 프레임 간격으로 고양이 이미지 추출
          # 프레임 0...3 -> image[0]
          # 프레임 4...7 -> image[1]
          # 프레임 8...11 -> image[2]
          # 프레임 12 -> image[3]
          self.image = self.images[self.frame//4]

      # 오른쪽 방향 이동 중인 경우
      if self.change_x > 0:
          # 프레임 카운터 증가
          self.frame += 1
          # 4 프레임 간격으로 고양이 이미지(images[4]~images[7]) 변경
          # image[7] 이 후의 경우(프레임이 12(3*4)보다 크면) 프레임 리셋
          if self.frame > 3*4:
              self.frame = 0
          # 매 4 프레임 간격으로 고양이 이미지 추출
          # 프레임 0...3 -> image[4]
          # 프레임 4...7 -> image[5]
          # 프레임 8...11 -> image[6]
          # 프레임 12 -> image[7]
          self.image = self.images[self.frame//4+4]
```



게임 예: 고양이 모험

플레이어 생성 및 초기화

```
pygame.init()

# 800x600 스크린  윈도우 설정
screen = pygame.display.set_mode([800, 600])
pygame.display.set_caption("고양이 모험")

# 고양이 플레이어 생성
player = Player( )
# 플레이어 초기 위치 설정
player.rect.x=50
player.rect.y=50
# 스프라이트 그룹 생성 후 플레이어  추가
movingsprites = pygame.sprite.Group()
movingsprites.add(player)

# 초기 배경 장면1 설정
current_room = 1
wall_list = setupRoomOne()

clock = pygame.time.Clock()
done = False
```



게임 예: 고양이 모험

이벤트 루프 – 키보드 이벤트

```
while done == False:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done=True

        # 키보드 누르면 이동속도 벡터 값 설정
        if event.type == pygame.KEYDOWN:
            # 왼쪽 화살표 키보드이면 이동속도 벡터 change_x = -5
            if event.key == pygame.K_LEFT:
                player.changespeed(-5,0)
            # 오른쪽 화살표 키보드이면 이동속도 벡터 change_x = 5
            if event.key == pygame.K_RIGHT:
                player.changespeed(5,0)
            # 위쪽 화살표 키보드이면 이동속도 벡터 change_y = -5
            if event.key == pygame.K_UP:
                player.changespeed(0,-5)
            # 아래쪽 화살표 키보드이면 이동속도 벡터 change_y = 5
            if event.key == pygame.K_DOWN:
                player.changespeed(0,5)
```

```
        # 키보드 떼면  이동속도 벡터 값 리셋
        # 이전 설정 값과 더해서 change_x = change_y = 0
        if event.type == pygame.KEYUP:
            if event.key == pygame.K_LEFT:
                player.changespeed(5,0)
            if event.key == pygame.K_RIGHT:
                player.changespeed(-5,0)
            if event.key == pygame.K_UP:
                player.changespeed(0,5)
            if event.key == pygame.K_DOWN:
                player.changespeed(0,-5)
```


게임 예: 고양이 모험

이벤트 루프 – 위치, 이미지, 배경화면 갱신

```
# 고양이 위치와 이미지 갱신
  player.update(wall_list)
  # 스크린 왼쪽 끝에 도달하면 배경 장면 변경
  if player.rect.x < -15:
      # 배경 장면1 => 배경 장면3
      if current_room == 1:
          wall_list = setupRoomThree()
          current_room = 3
          player.rect.x = 790
      # 배경 장면3 => 배경 장면2
      elif current_room == 3:
          wall_list = setupRoomTwo()
          player.rect.x = 790
          current_room = 2
      # 배경 장면2 => 배경 장면1
      else:
          wall_list = setupRoomOne()
          player.rect.x = 790
          current_room = 1

  # 스크린 오른쪽 끝에 도달하면 배경 장면 변경
  if player.rect.x > 801:
      # 배경 장면1 => 배경 장면2
      if current_room == 1:
          wall_list = setupRoomTwo()
          current_room = 2
          player.rect.x = 0
      # 배경 장면2 => 배경 장면3
      elif current_room == 2:
          wall_list = setupRoomThree()
          current_room = 3
          player.rect.x = 0
      # 배경 장면3 => 배경 장면1
      else:
          wall_list = setupRoomOne()
          current_room = 1
          player.rect.x = 0


  # 스크린 클리어
  screen.fill(back_color)

  # 고양이 플레이어 스프라이트 그리기
  movingsprites.draw(screen)
  # 배경 장면 그리기
  wall_list.draw(screen)

  # 스크린 갱신하여 디스플레
  pygame.display.flip()

  # 40 프레임/초
  clock.tick(40)
```


고양이 모험 코드

```
import pygame

black = [0,0,0]
white = [255,255,255]
back_color = [255,255,153]

# 장애물을 정의하는 클래스
# Sprite 부모 클래스를 상속
class GraphicWall(pygame.sprite.Sprite):
    # 이미지 추출 여러 개 복사 함수, 타일 위치/크기
    # 스크린 위치/크기 인수로 전달
    def setGraphic(self,tilex,tiley,tilewidth,tileheight,
                                          x,y, width,height):
        # 이미지 적재
        myimage =
       pygame.image.load("terrain_atlas.png").convert()
        # 스크린에 표시될 이미지 Surface 생성
        self.image = pygame.Surface([width, height])

        # 타일 크기 배수로 복사
        for row in range(height//tileheight):
            for column in range(width//tilewidth):
                self.image.blit(myimage,[column*tilewidth,
                  row*tileheight], [tilex,tiley,tilewidth,tileheight])
        # 이미지 크기의 rect 객체 지정
        self.rect = self.image.get_rect()
        # 장애물의 스크린 위치 지정
        self.rect.y = y
        self.rect.x = x
        # 투명키 설정
        self.image.set_colorkey(black)

    # 중간 부분이 확장된 타일 이미지 추출, 타일 위치 스크린 위
      치/크기 인수로 전달
    def setGraphic2(self,tilex,tiley,x,y,width,height):
        # 이미지 적재
        myimage =
       pygame.image.load("terrain_atlas.png").convert()
        # 스크린에 표시될 이미지 Surface 생성
        self.image = pygame.Surface([width, height])

        ## 1행 타일
        # 1행 1열 타일
        self.image.blit(myimage,[0,0],[tilex,tiley,32,32])
        # 1행 중간열 타일
        for column in range(width//32-2):
        self.image.blit(myimage,[(column+1)*32,0],
                                        [tilex+32,tiley,32,32])
```

```
        # 1행 마지막열 타일
        self.image.blit(myimage,[(width//32-1)*32,0],
                                        [tilex+64,tiley,32,32])

        ## 중간행 타일
        for row in range(height//32-2):
            self.image.blit(myimage,[0,(row+1)*32],
                                    [tilex,tiley+32,32,32])
            # row 열의 중간행 타일
            for column in range(width//32-2):
                self.image.blit(myimage, [(column+1)*32,
                    (row+1)*32], [tilex+32,tiley+32,32,32])
            # row 열의 마지막 행 타일
            self.image.blit(myimage,[(width//32-1)*32,(row+1)*32],
                                    [tilex+64,tiley+32,32,32])

        ## 마지막행 타일
        # 마지막행 1열 타일
        self.image.blit(myimage,[0,(height//32-1)*32],
                                        [tilex,tiley+64,32,32])
        # 마지막행 중간열 타일
        for column in range(width//32-2):
            self.image.blit(myimage,[(column+1)*32,
                    (height//32-1)*32],[tilex+32,tiley+64,32,32])
        # 마지막행 마지막열 타일
        self.image.blit(myimage,[(width//32-1)*32,
                    (height//32-1)*32],[tilex+64,tiley+64,32,32])

        # 이미지 크기의 rect 객체 지정
        self.rect = self.image.get_rect()
        # 장애물의 스크린 위치 지정
        self.rect.y = y
        self.rect.x = x
        # 투명키 설정
        self.image.set_colorkey(black)

# 석조 장애물 클래스
# GraphicWall 부모 클래스를 상속
class StoneWall(GraphicWall):
    # 생성자, 스크린 위치/크기 인수로 전달
    def __init__(self,x,y,width,height):
        # 부모 클래스 생성자 호출
        pygame.sprite.Sprite.__init__(self)
        # 타일 이미지 위치
        tilex=32*16
        tiley=32*29
        # 중간 확장 이미지 추출
        self.setGraphic2(tilex,tiley,x,y,width,height)

# 벽돌 장애물 클래스
# GraphicWall 부모 클래스를 상속
class StoneWall2(GraphicWall):
    # 생성자, 스크린 위치/크기 인수로 전달
    def __init__(self,x,y,width,height):
        # 부모 클래스 생성자 호출
        pygame.sprite.Sprite.__init__(self)
        # 타일 이미지 위치
        tilex=32*16
        tiley=32*23
        # 중간 확장 이미지 추출
        self.setGraphic2(tilex,tiley,x,y,width,height)
```

```
# 나무1 클래스
# GraphicWall 부모 클래스를 상속
class Tree1(GraphicWall):
    # 생성자, 스크린 위치 인수로 전달
    def __init__(self,x,y):
        # 부모 클래스 생성자 호출
        pygame.sprite.Sprite.__init__(self)
        # 스크린 이미지 크기
        width=64
        height=160
        # 타일 이미지 크기
        tilewidth=64
        tileheight=160
        # 타일 이미지 위치
        tilex=32*30
        tiley=32*0
        # 이미지 추출
        self.setGraphic(tilex,tiley,tilewidth,tileheight,x,y,width,height)

# 나무2 클래스
# GraphicWall 부모 클래스를 상속
class Tree2(GraphicWall):
    # 생성자, 스크린 위치 인수로 전달
    def __init__(self,x,y):
        # 부모 클래스 생성자 호출
        pygame.sprite.Sprite.__init__(self)
        # 스크린 이미지 크기
        width=96
        height=128*2        # 타일 2배 높이 지정
        # 타일 이미지 크기
        tilewidth=96
        tileheight=128
        # 타일 이미지 위치
        tilex=32*29
        tiley=32*28
        # 이미지 추출
        self.setGraphic(tilex,tiley,tilewidth,tileheight,x,y,width,
        height)

# 물 웅덩이 클래스
# GraphicWall 부모 클래스를 상속
class WaterWall(GraphicWall):
    # 생성자, 스크린 위치/크기 인수로 전달
    def __init__(self,x,y,width,height):
        # 부모 클래스 생성자 호출
        pygame.sprite.Sprite.__init__(self)
        # 타일 이미지 위치
        tilex=32*9
        tiley=32*11
        # 중간 확장 이미지 추출
        self.setGraphic2(tilex,tiley,x,y,width,height)

# 가든 장벽 클래스
# GraphicWall 부모 클래스를 상속
class GardenWall(GraphicWall):
    # 생성자, 스크린 위치/크기 인수로 전달
    def __init__(self,x,y,width,height):
        # 부모 클래스 생성자 호출
        pygame.sprite.Sprite.__init__(self)
```

```
        # 타일 이미지 위치
        tilex=32*5
        tiley=32*17
        # 중간 확장 이미지 추출
        self.setGraphic2(tilex,tiley,x,y,width,height)

# 푸른 잔디 장벽 클래스
# GraphicWall 부모 클래스를 상속
class TallGrass(GraphicWall):
    # 생성자, 스크린 위치/크기 인수로 전달
    def __init__(self,x,y,width,height):
        # 부모 클래스 생성자 호출
        pygame.sprite.Sprite.__init__(self)
        # 타일 이미지 위치
        tilex=32*0
        tiley=32*22
        # 중간 확장 이미지 추출
        self.setGraphic2(tilex,tiley,x,y,width,height)

# 노란 잔디 장벽 클래스
# GraphicWall 부모 클래스를 상속
class FallGrass(GraphicWall):
    # 생성자, 스크린 위치/크기 인수로 전달
    def __init__(self,x,y,width,height):
        # 부모 클래스 생성자 호출
        pygame.sprite.Sprite.__init__(self)
        # 타일 이미지 위치
        tilex=32*0
        tiley=32*28
        # 중간 확장 이미지 추출
        self.setGraphic2(tilex,tiley,x,y,width,height)
```

```
# 플레이어 고양이를 정의하는 클래스
# Sprite 부모 클래스를 상속
class Player(pygame.sprite.Sprite):
    # 클래스 속성
    # 고양이 이동속도 벡터 값 초기화, 왼쪽(음수) 오른쪽(양수) 방향
    change_x=0
    change_y=0
    # 고양이 이미지 선택을 위한 프레임 카운터
    # 4개 프레임 마다 이미지 선택
    frame = 0
    # 생성자
    def __init__(self):
        # 부모 클래스 생성자 호출
        pygame.sprite.Sprite.__init__(self)
        # 고양이 이미지 리스트 초기화
        self.images=[]
        # cat1.png ~ cat8.png 고양이 이미지 적재
        for i in range(1,9):
            img = pygame.image.load("cat"+str(i)+".png").convert()
            img.set_colorkey(white)
            self.images.append(img)
        # 디폴트로 고양이 images[0] 사용
        self.image = self.images[0]
        # 이미지 크기의 rect 객체 지정
        self.rect = self.image.get_rect()

    # 이동속도 벡터 값 설정
    def changespeed(self,x,y):
        self.change_x+=x
        self.change_y+=y
```

```
    # 고양이 위치와 이미지 갱신
    def update(self, walls):
        # x 축 새 위치 설정, 충돌 시 복구를 위해  이전 위치 저장
        old_x=self.rect.x
        new_x=old_x+self.change_x
        self.rect.x = new_x
        # x 방향 이동 시 충돌 여부 감지
        collide = pygame.sprite.spritecollide(self, walls, False)
        if collide:
            # 충돌인 경우 이 전 위치 복구
            self.rect.x=old_x
        # y 축 새 위치 설정, 충돌 시 복구를 위해  이전 위치 저장
        old_y=self.rect.y
        new_y=old_y+self.change_y
        self.rect.y = new_y
        # y 방향 이동 시 충돌 여부 감지
        collide = pygame.sprite.spritecollide(self, walls, False)
        if collide:
            # 충돌인 경우 이전 위치 복구
            self.rect.y=old_y

        # 왼쪽 방향 이동 중인 경우
        if self.change_x < 0:
            # 프레임 카운터 증가
            self.frame += 1
            # 4 프레임 간격으로 고양이 이미지
             # (images[0]~images[3]) 변경
            # image[3] 이 후의 경우(프레임이 12(3*4)보다 크면)
            # 프레임 리셋
            if self.frame > 3*4:
                self.frame = 0
            # 매 4 프레임 간격으로 고양이 이미지 추출
            # 프레임 0...3 -> image[0]
            # 프레임 4...7 -> image[1]
            # 프레임 8...11 -> image[2]
            # 프레임 12 -> image[3]
            self.image = self.images[self.frame//4]

        # 오른쪽 방향 이동 중인 경우
        if self.change_x > 0:
            # 프레임 카운터 증가
            self.frame += 1
            # 4 프레임 간격으로 고양이 이미지
            # (images[4]~images[7]) 변경
            # image[7] 이 후의 경우(프레임이 12(3*4)보다 크면)
            # 프레임 리셋
            if self.frame > 3*4:
                self.frame = 0
            # 매 4 프레임 간격으로 고양이 이미지 추출
            # 프레임 0...3 -> image[4]
            # 프레임 4...7 -> image[5]
            # 프레임 8...11 -> image[6]
            # 프레임 12 -> image[7]
            self.image = self.images[self.frame//4+4]
```

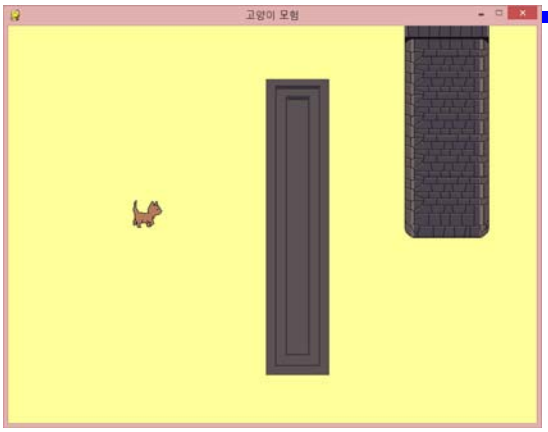
```
# 배경 장면1
def setupRoomOne():
    # 스프라이트 그룹 생성
    wall_list=pygame.sprite.Group()

    # 석조 장애물 추가
    wall=StoneWall(390,80, 96, 448)
    wall_list.add(wall)
    # 벽돌 장애물
    wall=StoneWall2(600,0, 128, 320)
    wall_list.add(wall)

    # 스프라이트 그룹 리턴
    return wall_list

# 배경 장면2
def setupRoomTwo():
    # 스프라이트 그룹 생성
    wall_list=pygame.sprite.Group()

    # 나무1 추가
    wall=Tree1(100,100)
    wall_list.add(wall)
    # 나무2 추가
    wall=Tree2(300,250)
    wall_list.add(wall)
    # 잔디 장벽 추가
    wall=TallGrass(100,400,64,160)
    wall_list.add(wall)
    # 나무1 추가
    wall=Tree1(500,160)
    wall_list.add(wall)
    # 노란 잔디 장벽 추가
    wall=FallGrass(600,128,192,160)
    wall_list.add(wall)
    # 나무1 추가
    wall=Tree1(700,350)
    wall_list.add(wall)

    # 스프라이트 그룹 리턴
    return wall_list

# 배경 장면3
def setupRoomThree():
    # 스프라이트 그룹 생성
    wall_list=pygame.sprite.Group()

    # 물 웅덩이 추가
    wall=WaterWall(64,64,192,192)
    wall_list.add(wall)
    # 가든 장벽 추가
    wall=GardenWall(128,325,256,192)
    wall_list.add(wall)
    # 나무1 추가
    wall=Tree1(520,256)
    wall_list.add(wall)

    # 스프라이트 그룹 리턴
    return wall_list
```

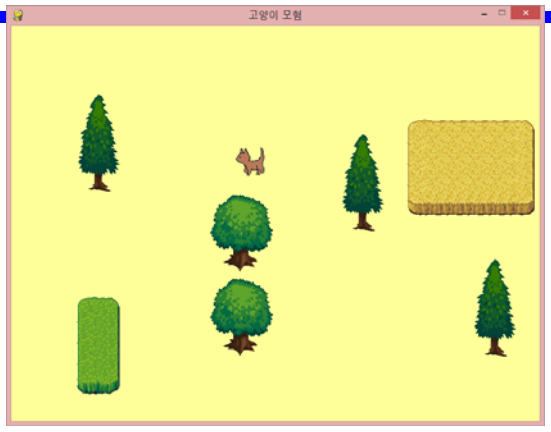
```
pygame.init()

# 800x600 스크린  윈도우 설정
screen = pygame.display.set_mode([800, 600])
pygame.display.set_caption("고양이 모험")

# 고양이 플레이어 생성
player = Player( )
# 플레이어 초기 위치 설정
player.rect.x=50
player.rect.y=50
# 스프라이트 그룹 생성 후 플레이어  추가
movingsprites = pygame.sprite.Group()
movingsprites.add(player)

# 초기 배경 장면1 설정
current_room = 1
wall_list = setupRoomOne()

clock = pygame.time.Clock()
done = False

while done == False:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done=True

        # 키보드 누르면 이동속도 벡터 값 설정
        if event.type == pygame.KEYDOWN:
            # 왼쪽 화살표 키보드이면 이동속도 벡터 change_x = -5
            if event.key == pygame.K_LEFT:
                player.changespeed(-5,0)
            # 오른쪽 화살표 키보드이면 이동속도 벡터 change_x = 5
            if event.key == pygame.K_RIGHT:
                player.changespeed(5,0)
            # 위쪽 화살표 키보드이면 이동속도 벡터 change_y = -5
            if event.key == pygame.K_UP:
                player.changespeed(0,-5)
            # 아래쪽 화살표 키보드이면 이동속도 벡터 change_y = 5
            if event.key == pygame.K_DOWN:
                player.changespeed(0,5)

        # 키보드 떼면  이동속도 벡터 값 리셋
        # 이전 설정 값과 더해서 change_x = change_y = 0
        if event.type == pygame.KEYUP:
            if event.key == pygame.K_LEFT:
                player.changespeed(5,0)
            if event.key == pygame.K_RIGHT:
                player.changespeed(-5,0)
            if event.key == pygame.K_UP:
                player.changespeed(0,5)
            if event.key == pygame.K_DOWN:
                player.changespeed(0,-5)

    # 고양이 위치와 이미지 갱신
    player.update(wall_list)
```

```
    # 스크린 왼쪽 끝에 도달하면 배경 장면 변경
    if player.rect.x < -15:
        # 배경 장면1 => 배경 장면3
        if current_room == 1:
            wall_list = setupRoomThree()
            current_room = 3
            player.rect.x = 790
        # 배경 장면3 => 배경 장면2
        elif current_room == 3:
            wall_list = setupRoomTwo()
            player.rect.x = 790
            current_room = 2
        # 배경 장면2 => 배경 장면1
        else:
            wall_list = setupRoomOne()
            player.rect.x = 790
            current_room = 1

    # 스크린 오른쪽 끝에 도달하면 배경 장면 변경
    if player.rect.x > 801:
        # 배경 장면1 => 배경 장면2
        if current_room == 1:
            wall_list = setupRoomTwo()
            current_room = 2
            player.rect.x = 0
        # 배경 장면2 => 배경 장면3
        elif current_room == 2:
            wall_list = setupRoomThree()
            current_room = 3
            player.rect.x = 0
```

```
        # 배경 장면3 => 배경 장면1
        else:
            wall_list = setupRoomOne()
            current_room = 1
            player.rect.x = 0


    # 스크린 클리어
    screen.fill(back_color)

    # 고양이 플레이어 스프라이트 그리기
    movingsprites.draw(screen)
    # 배경 장면 그리기
    wall_list.draw(screen)

    # 스크린 갱신하여 디스플레
    pygame.display.flip()

    # 40 프레임/초
    clock.tick(40)

pygame.quit()
```


시험주행





과제

1. 지금까지 배운 내용을 활용한 텀 프로젝트 부분 프로그램 작성
   - 텀 프로젝트 계획 설명
   - 지금까지 배운 내용 활용한 부분 프로그램 소스
   - 실행 결과