Project Progress Report

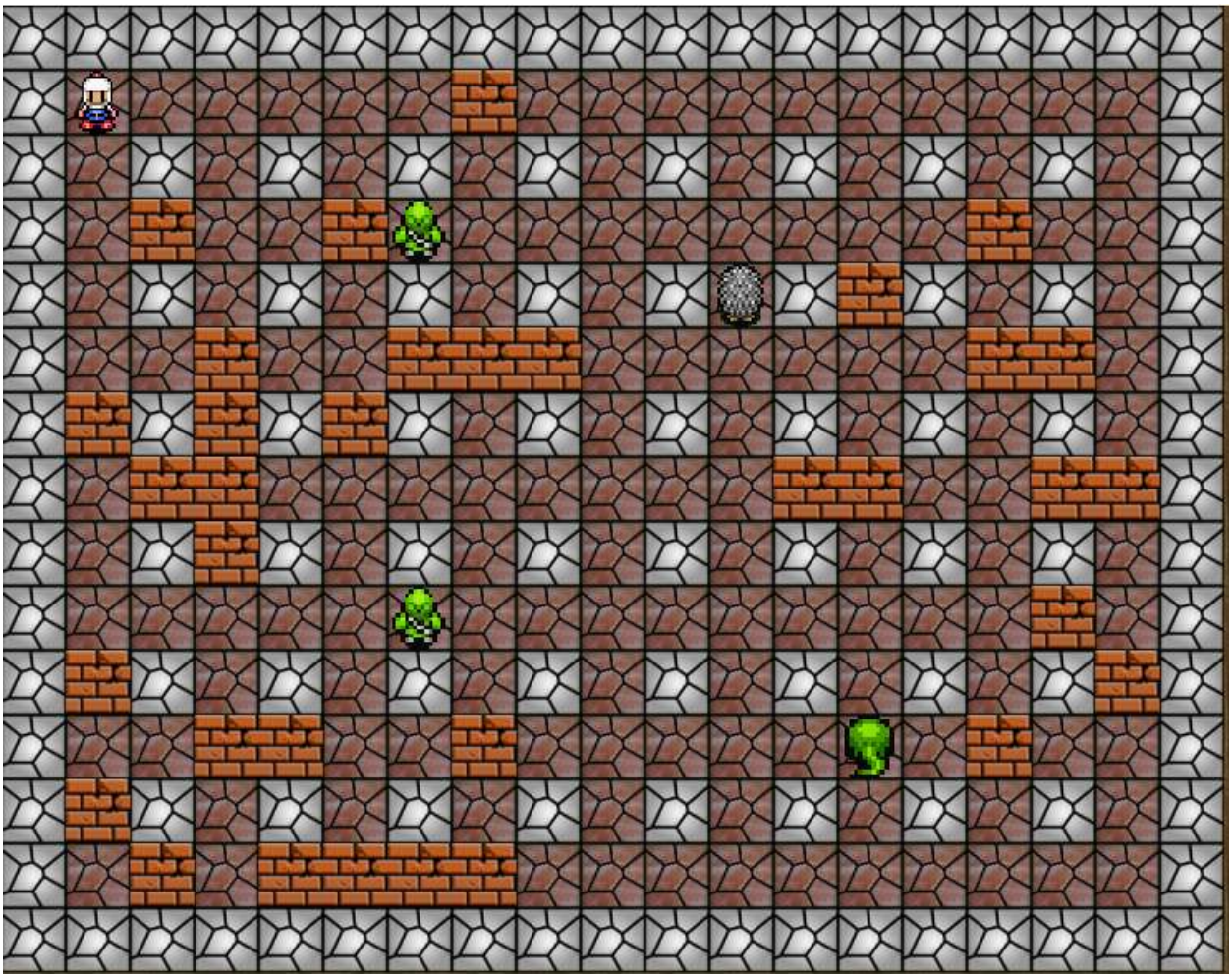
Name: Zhiwei Xin

Andrew Id: zxin

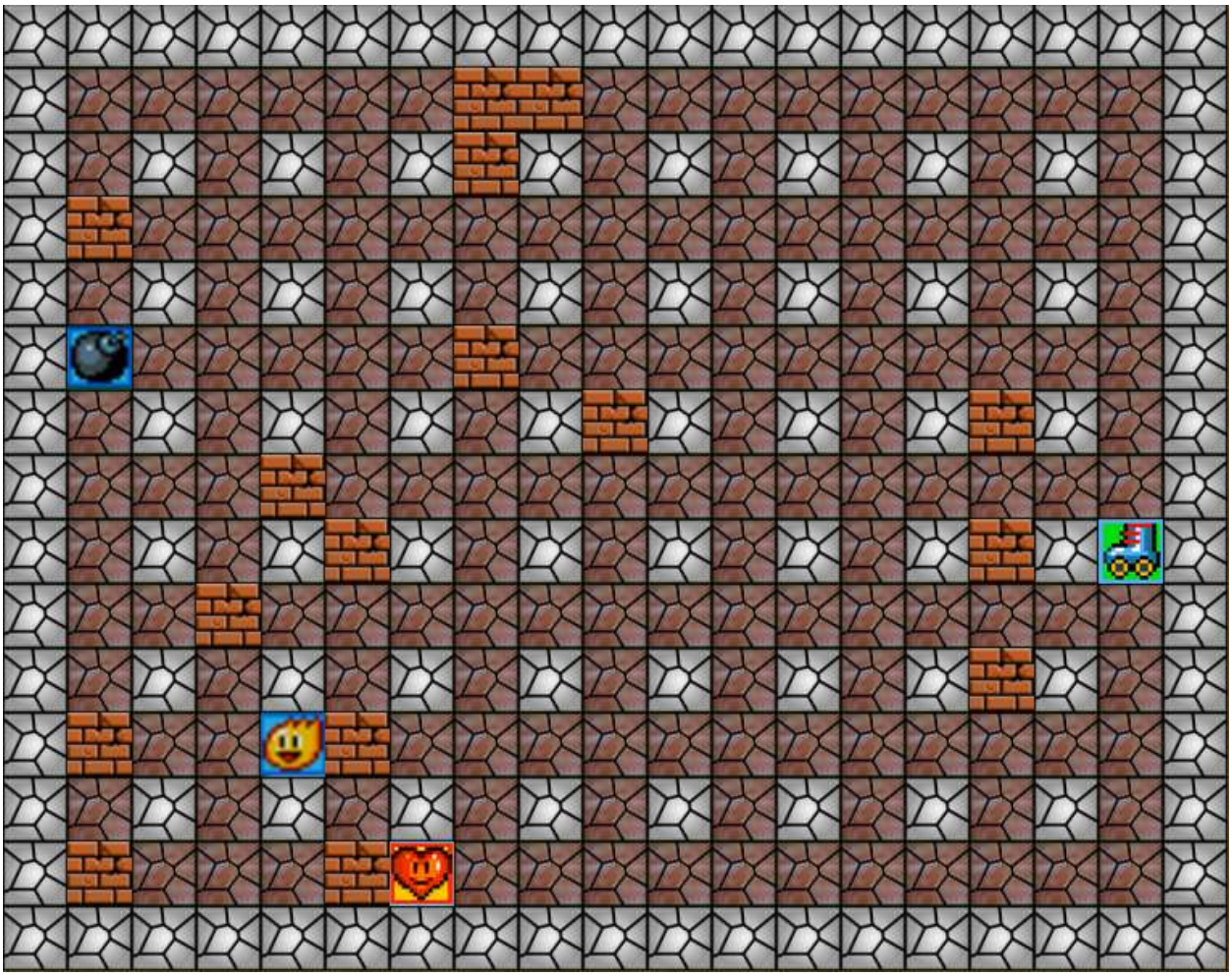
Brief Description

The game can basically run. Most resources needed have been found, but most features are unimplemented.

Screenshots



We can move the characters and the monsters can move itself, but collisions are not implemented



Code snippets

I can provide the python files, but it seems code snippets are preferred.

If need python file, please contact me.

```
# This represents the bomb class.
import configure
import pygame


class Bomb(pygame.sprite.Sprite):
    def __init__(self, player):
        pygame.init()
        pygame.sprite.Sprite.__init__(self)
        self.c = configure.Configure()
        imagePath = self.c.IMAGE_PATH + "/bomb.png"
        self.image = pygame.image.load(imagePath).convert()
```

```
        self.range = player.power  # kill zone
        self.triggered = False
        self.countDown = 3  # how many seconds
        self.owner = player
        self.position = self.image.get_rect()
        self.position = self.position.move((player.position.x,
                                            player.position.y))
        tsize = self.c.TILE_SIZE
        self.rect = pygame.Rect(self.position[0],
self.position[1], tsize, tsize)

    def countdown(self):
        self.countDown -= 1

    def explode(self):
        self.owner.curBombs += 1
        # shall i add animation here
```

```
# The Character Class

import configure
import copy
import pygame


class Character(pygame.sprite.Sprite):
    """docstring for Character"""
    def __init__(self, idNum, position, thisType):
        self.c = configure.Configure() # some constant
        pygame.sprite.Sprite.__init__(self)
        # super(Character,self).__init__() # Modified
        pygame.init()
        self.id = idNum

        self.type = thisType # players/ enemies

        size = (self.c.WIDTH, self.c.HEIGHT)
        self.screen = pygame.display.set_mode(size)

        self.getImage("down")

        self.rect = self.image.get_rect()
        self.position = self.image.get_rect()
        self.move(position)

    def getImage(self,direction):
        self.direction = direction
```

```
        if self.type == "players":
            imagePath = self.c.IMAGE_PATH + "players/"
+"p_"+str(self.id)+"_" + direction + ".png"
        else:
            imagePath = self.c.IMAGE_PATH + "enemies/"
+"e_"+str(self.id)+"_" + direction + ".png"

        self.image = pygame.image.load(imagePath).convert()

        # solve the face to
    def face(self,key):
        self.getImage(key)

    def moveSquare(self,key):
        if key == pygame.K_RIGHT:
            self.face("right")
            return (1*self.c.TILE_SIZE, 0)
        elif key == pygame.K_LEFT:
            self.face("left")
            return (-1*self.c.TILE_SIZE, 0)
        elif key == pygame.K_UP:
            self.face("up")
            return (0, -1*self.c.TILE_SIZE)
        elif key == pygame.K_DOWN:
            self.face("down")
            return (0, 1 * self.c.TILE_SIZE)


    def move(self,point):
        self.old = copy.copy(self.rect)
        dx,dy = point[0],point[1]
        self.rect.x,self.rect.y = self.rect.x +
dx,self.rect.y+dy
        self.position = self.position.move(point)



    def updateRect(self):
        tSize = self.c.TILE_SIZE


    def update(self):
        self.updateRect()

# --------- test code ------------ #
    def innerRun(self):
        pygame.init()
```

```
        size = (700, 500)
        screen = pygame.display.set_mode(size)

        user = Character(1,(40,40),"players")
        self.userGroup = pygame.sprite.Group(user)
        # self.userGroup.add(self.user)

        # self.screen.blit(self.image.convert(),[0,0])
        pygame.display.set_caption("My Test")

        # Loop until the user clicks the close button.
        done = False

        # Used to manage how fast the screen updates
        clock = pygame.time.Clock()

        # -------- Main Program Loop -----------
        while not done:
            # --- Main event loop
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    done = True
                elif event.type == pygame.KEYDOWN:
                    if event.key in
[pygame.K_LEFT,pygame.K_RIGHT,pygame.K_UP,pygame.K_DOWN]:
                        point = user.moveSquare(event.key)
                        user.move(point)

            screen.fill((255,255,255))
            self.userGroup.draw(screen)
            # --- Game logic should go here

            # --- Drawing code should go here

            # First, clear the screen to white. Don't put other
drawing commands
            # above this, or they will be erased with this
command.

            # --- Go ahead and update the screen with what
we've drawn.
            pygame.display.flip()

            # --- Limit to 60 frames per second
            clock.tick(30)

        # Close the window and quit.
```

```
        # If you forget this line, the program will 'hang'
        # on exit if running from IDLE.
        pygame.quit()
```

```
# enemy
import pygame, character, random


class Enemy(character.Character):
    """docstring for Enemy"""

    def __init__(self, idNum, position, thisType):
        # super().__init__(idNum,position,thisType)
        character.Character.__init__(self, idNum, position,
thisType)

    def movement(self):
        patterns = [pygame.K_UP, pygame.K_DOWN, pygame.K_LEFT,
pygame.K_RIGHT]

        # this move pattern can be improved
        return self.moveSquare(random.choice(patterns))
```

```
# The Game Menu


import pygame, configure, instructions, game, pickle, maps,
costumizedLevel
import highscore

# constant
PTS_WIDTH = 1024  # pre screen width
PTS_HEIGHT = 768  # pre Screen height


class TitleScreen(object):
    def __init__(self):
        self.c = configure.Configure()
        done = False
        pygame.init()
        clock = pygame.time.Clock()
        self.modifyMap = False
        wantToEnd = False
        while not done:
            self.screen = pygame.display.set_mode([PTS_WIDTH,
PTS_HEIGHT])
            pygame.display.set_caption("Bomberman")
```

```
            # import background image
            bgImagePath = self.c.IMAGE_PATH + "titleScreen.png"
            bgImage = pygame.image.load(bgImagePath).convert()
            bgImage = pygame.transform.scale(bgImage,
(PTS_WIDTH, PTS_HEIGHT))
            self.screen.blit(bgImage, [0, 0])

            pygame.mixer.music.load(self.c.AUDIO_PATH +
"title.mid")
            pygame.mixer.music.play()

            notValidOp = False
            # under valid control mode
            while not notValidOp:
                # get mouse position
                pos = pygame.mouse.get_pos()
                # testCode
                for event in pygame.event.get():
                    # deal with the exit
                    if event.type == pygame.QUIT:
                        notValidOp = not notValidOp
                        done = not done
                        wantToEnd = True

                    # load the game
                    elif event.type == pygame.KEYDOWN:
                        if event.key == pygame.K_l:
                            with open("saved.pickle", "rb") as
f:
                                self.total = pickle.load(f)
                            f.close()
                            self.playGame(self.c.SINGLE,
self.total)

                    elif event.type == pygame.MOUSEBUTTONDOWN:
                        # map editor
                        if self.inBoundary(pos[0], pos[1], 25,
450, 250, 500):
                            # customize the map
                            costumizedLevel.CustomizedLevel()
                            # the following code are necessary
for python3
                            notValidOp = not notValidOp
                            # it should work like above, when
finished one op
                            # it should jump the loop so the
```

```
video and music
                            # can be reinitialized again.
                        # single
                        elif self.inBoundary(pos[0], pos[1],
25, 500, 250, 550):
                            pygame.mixer.music.fadeout(1000)
                            self.playGame(self.c.SINGLE, None)
                            # following code are necessary for
python 3
                            notValidOp = not notValidOp

                        # multi
                        elif self.inBoundary(pos[0], pos[1],
25, 550, 250, 600):
                            pygame.mixer.music.fadeout(1000)
                            self.playGame(self.c.MULTI, None)
                            # the following code are necessary
for python3
                            notValidOp = not notValidOp

                        # instructions
                        elif self.inBoundary(pos[0], pos[1],
25, 600, 250, 650):
                            self.instructions()

                            notValidOp = not notValidOp
                            # it should work like above, when
finished one op
                            # it should jump the loop so the
video and music
                            # can be reinitialzed again.

                        # high Score
                        elif self.inBoundary(pos[0], pos[1],
25, 650, 250, 700):
                            self.highScores()

                            notValidOp = not notValidOp
                            # it should work like above, when
finished one op
                            # it should jump the loop so the
video and music
                            # can be reinitialzed again.

                        # exit
                        elif self.inBoundary(pos[0], pos[1],
40, 700, 250, 750):
```

```
                            done = not done
                            notValidOp = not notValidOp
                            # wantToEnd = True

                # Go ahead and update the screen with what
we've drawn.

                pygame.display.flip()

                # Limit to 60 frames per second
                clock.tick(self.c.FPS)

        pygame.quit()

    def inBoundary(self, x0, y0, x1, y1, x2, y2):
        if (x1 <= x0 <= x2) and (y1 <= y0 <= y2):
            return True
        return False

    def instructions(self):
        instructions.Instructions()

    def playGame(self, mode, saved):
        game.Game(mode, saved)

    def highScores(self):
        hs = highscore.HighScores()
        hs.displayScores()
```