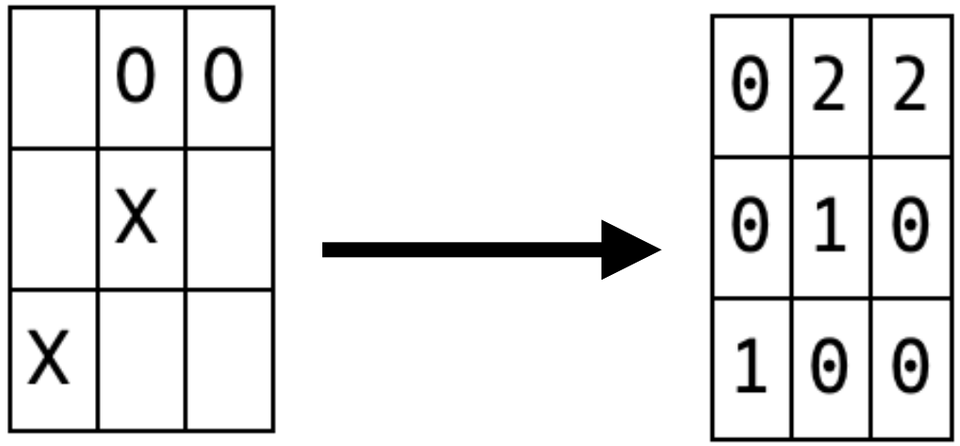
http://www.pygame.org/

PYGAME

Mapas

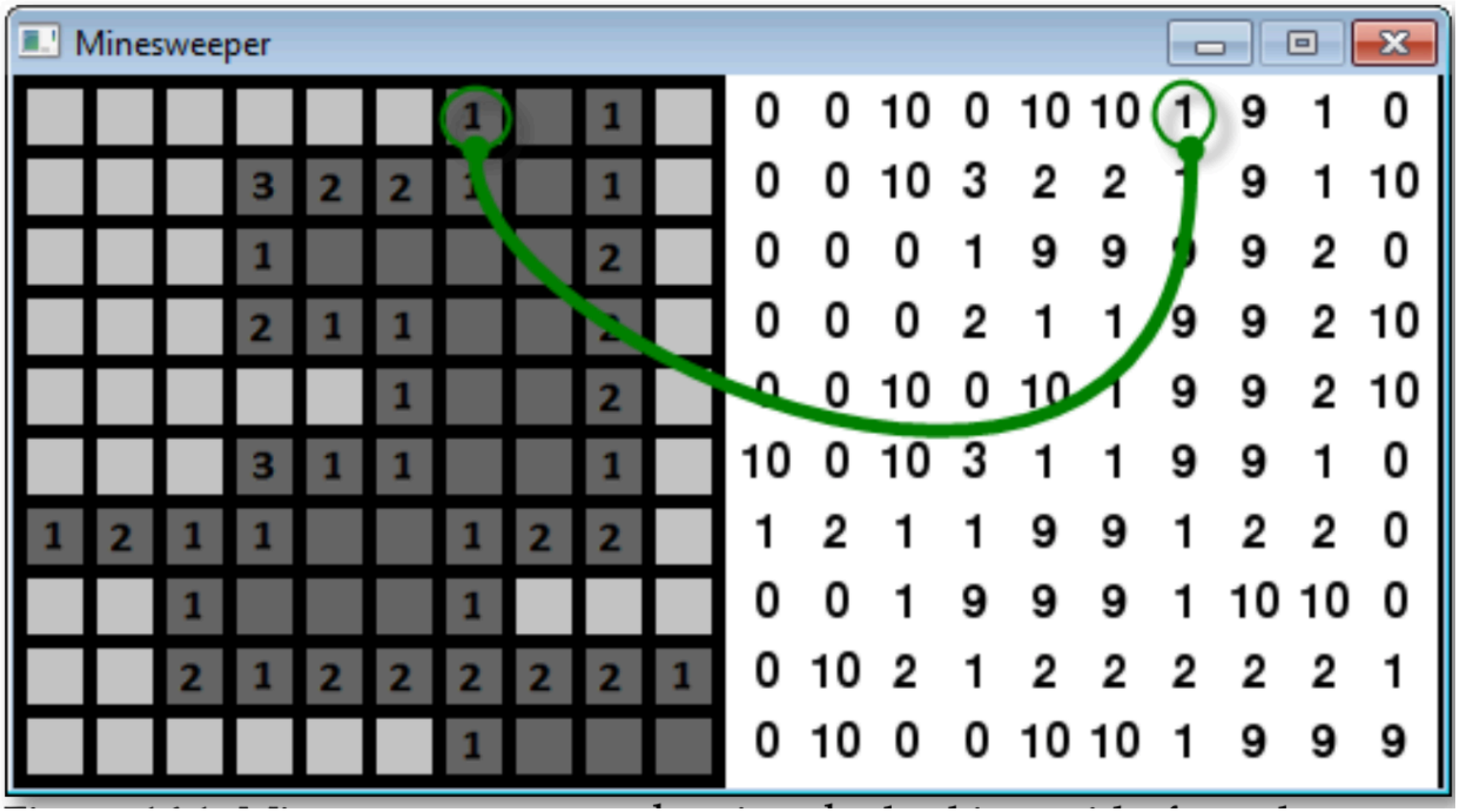
1) Grid (Tabuleiro)

- Muitos jogos como campo minado, xadrez ou jogo da velha fazem uso de uma matriz (grid) de números (vetor bidimensional).



Exemplo do Grid de Campo Minado

- O número 10 representa uma mina
- O número 0 representa um espaço em branco que não foi clicado
- O número 9 é um espaço vazio não clicado
- Os números 1 a 8 representam quantas minas existem nos 8 quadrados da volta e só é mostrado quando o usuário clica nele.



Podemos ter uma segunda matriz com as flags.

Tiled Map

- Jogos clássicos de aventura usam mapas que são criados a partir de um Tiled Map Editor (http://www.mapeditor.org/)

Tiled Map

- Cada espaço do mapa (matriz) é representado por um número, indicando o tipo de terreno: terra, pedra, grama, etc.

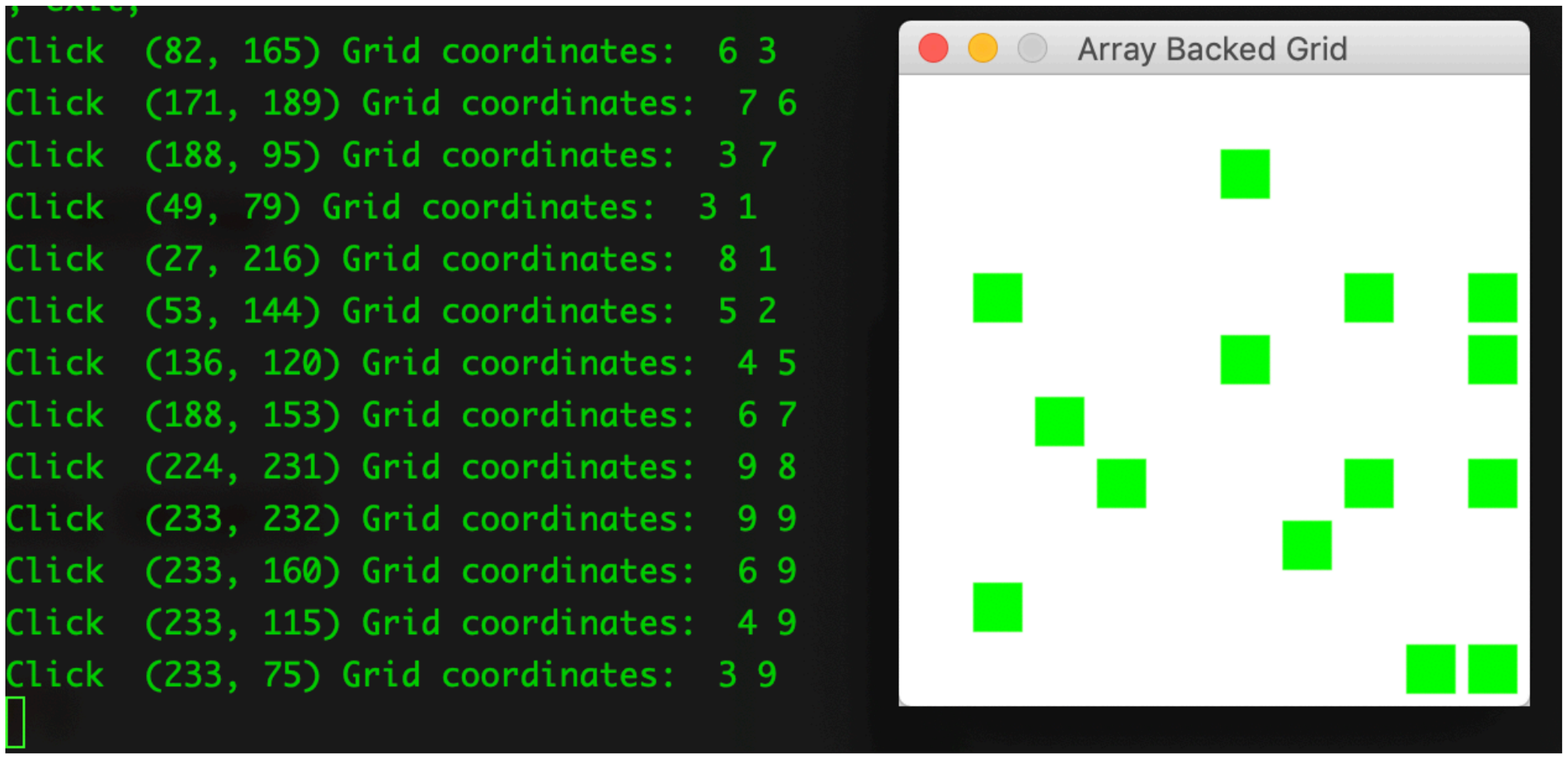
- Como carregar mapas em Python:

https://github.com/bitcraft/PyTMX

```
from pytmx.util_pygame import load_pygame
tiled_map = load_pygame('map.tmx')
```

Exemplo de Mapa

- Criaremos um grid de 255 x 255 elementos no qual cada vez que clicarmos nele, o campo fica verde.



```
12  import pygame
13
14  # Define some colors
15  BLACK = (0, 0, 0)
16  WHITE = (255, 255, 255)
17  GREEN = (0, 255, 0)
18  RED = (255, 0, 0)
19
20  # This sets the WIDTH and HEIGHT of each grid location
21  WIDTH = 20
22  HEIGHT = 20
23  # This sets the margin between each cell
24  MARGIN = 5
25
26  def cria_matriz():
27      #Alternative way to create the grid (10 x 10)
28      #grid = [[0 for x in range(10)] for y in range(10)]
29
30      # --- Create grid of numbers
31      # Create an empty list
32      grid = []
33      # Loop for each row
34      for row in range(WIDTH):
35          # For each row, create a list that will
36          # represent an entire row
37          grid.append([])
38          # Loop for each column
39          for column in range(HEIGHT):
40              # Add a the number zero to the current row
41              grid[row].append(0)
42      return grid
```

Definimos os parâmetros gerais

Criamos a matriz inicializando com zeros

```
44  pygame.init()
45
46  # Set the width and height of the screen [width, height]
47  size = (255, 255)
48  screen = pygame.display.set_mode(size)
49
50  pygame.display.set_caption("Array Backed Grid")
51
52  # Loop until the user clicks the close button.
53  done = False
54
55  # Used to manage how fast the screen updates
56  clock = pygame.time.Clock()
57
58  grid = cria_matriz()
59   # Set row 1, cell 5 to one. (Remember rows and
60  # column numbers start at zero.)
61  grid[1][5] = 1
62
63   # -------- Main Program Loop -----------
64  while not done:
65      # --- Main event loop
66      for event in pygame.event.get():
67          if event.type == pygame.QUIT:
68              done = True
69          elif event.type == pygame.MOUSEBUTTONDOWN:
70              # User clicks the mouse. Get the position
71              pos = pygame.mouse.get_pos()
72              # Change the x/y screen coordinates to grid coordinates
73              column = pos[0] // (WIDTH + MARGIN)
74              row = pos[1] // (HEIGHT + MARGIN)
75              # Set that location to one
76              grid[row][column] = 1
77              print("Click ", pos, "Grid coordinates: ", row, column)
78      # --- Game logic should go here
```

Criamos a matriz e marcamos o **1 x 5**

Tratamos os eventos de mouse

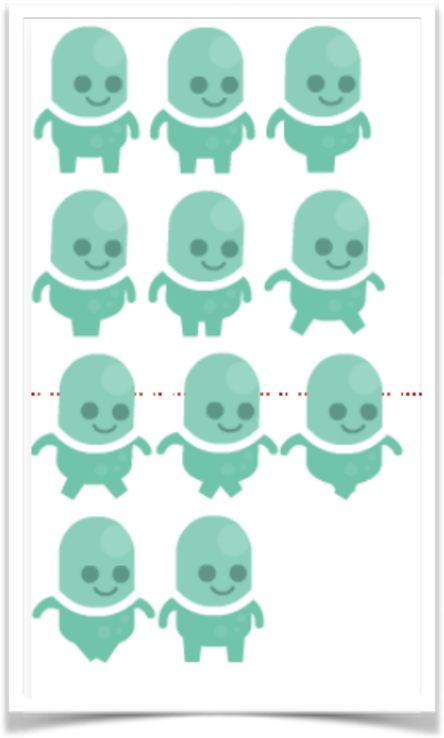
```
78      # --- Game logic should go here
79
80      # --- Screen-clearing code goes here
81      # If you want a background image, replace this clear with blit'ing the
82      # background image.
83      screen.fill(WHITE)
84
85      # --- Drawing code should go here
86      #Draw the grid
87      for row in range(WIDTH):
88          for column in range(HEIGHT):
89              color = WHITE
90              if grid[row][column] == 1:
91                  color = GREEN
92              pygame.draw.rect(screen,
93                               color,
94                               [(MARGIN + WIDTH) * column + MARGIN,
95                                (MARGIN + HEIGHT) * row + MARGIN,
96                                WIDTH,
97                                HEIGHT])
98
99      # --- Go ahead and update the screen with what we've drawn.
100     pygame.display.flip()
101
102     # --- Limit to 60 frames per second
103     clock.tick(60)
104
105 # Close the window and quit.
106 pygame.quit()
```

Desenha o Grid

2) Jogo de Plataforma

- Quando precisamos representar um personagem que irá se movimentar em um jogo 2D, precisamos de um *sprite sheet*.

- Ele é uma imagem que mostra todas as possíveis movimentações de um personagem.

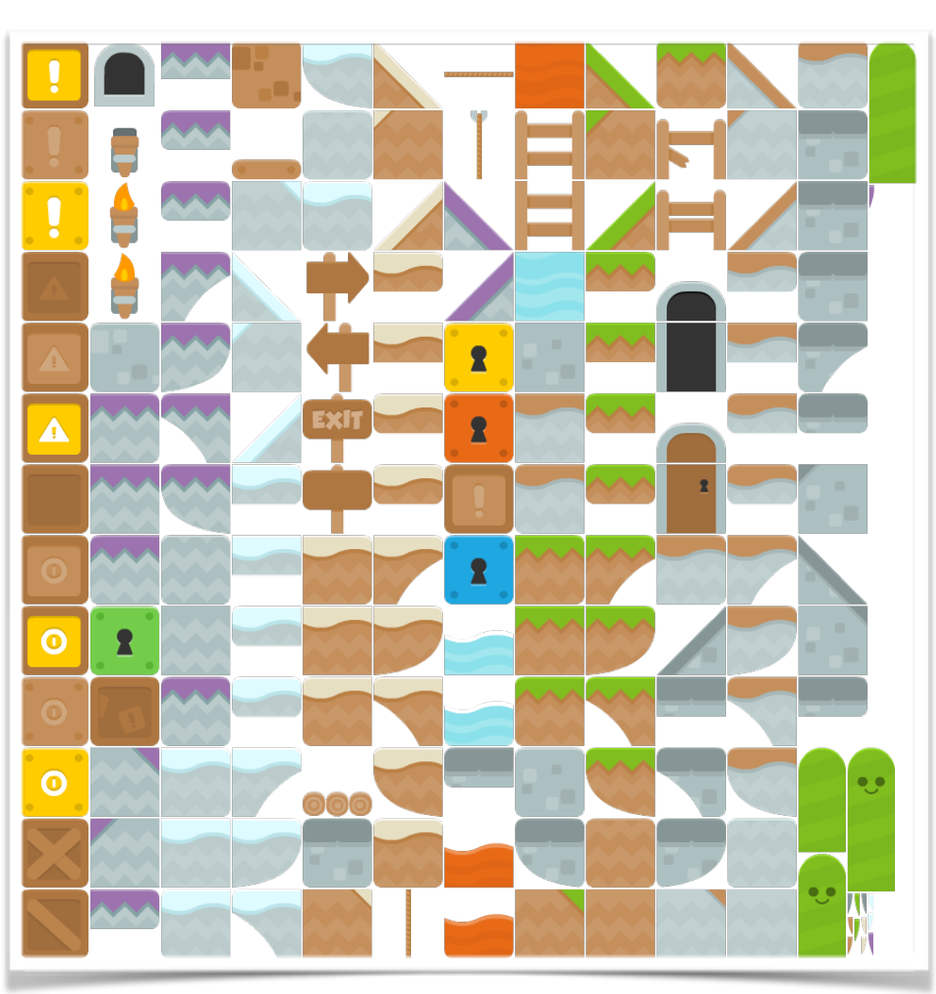


Sprite Sheets

- O exemplo anterior foi baixado do site:

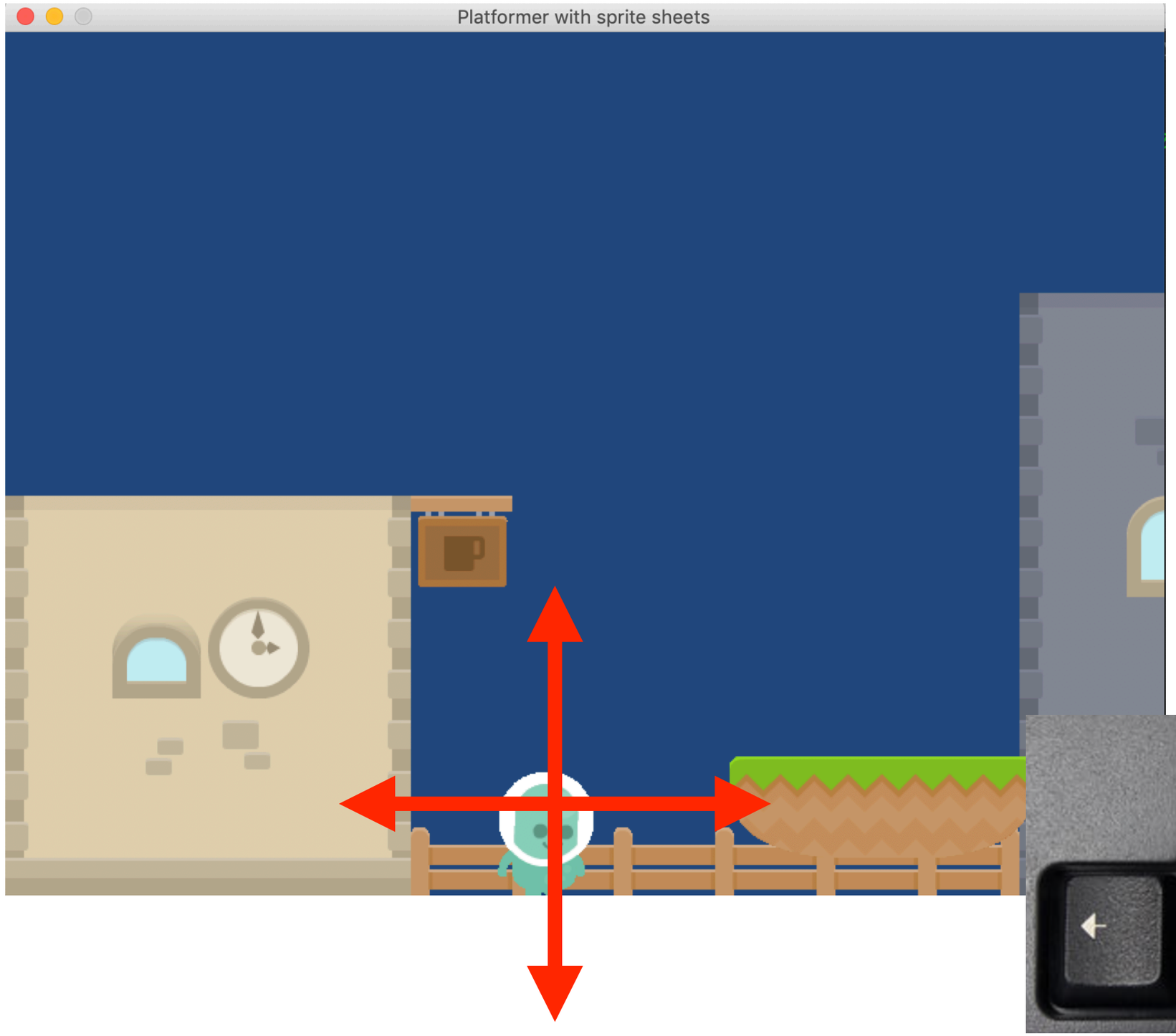
http://opengameart.org/content/platformer-art-deluxe

- Podemos ter um Sprite Sheet que representa o mapa:



Exemplo de Sprite Sheets

- Criaremos um jogo no qual controlamos o personagem com as setas do teclado.



```
34  def main():
35      """ Main Program """
36      pygame.init()
37
38      # Set the height and width of the screen
39      size = [constants.SCREEN_WIDTH, constants.SCREEN_HEIGHT]
40      screen = pygame.display.set_mode(size)
41
42      pygame.display.set_caption
43
44      # Create the player
45      player = Player()
46
47      # Create all the levels
48      level_list = []
49      level_list.append(levels.Level_01(player))
50      level_list.append(levels.Level_02(player))
51
52      # Set the current level
53      current_level_no = 0
54      current_level = level_list[current_level_no]
55
56      active_sprite_list = pygame.sprite.Group()
57      player.level = current_level
58
59      player.rect.x = 340
60      player.rect.y = constants.SCREEN_HEIGHT - player.rect.height
61      active_sprite_list.add(player)
62
63      #Loop until the user clicks the close button.
64      done = False
65
66      # Used to manage how fast the screen updates
67      clock = pygame.time.Clock()
```

Cria o Jogador e os Níveis

platform_scroller.py

```
34  def main():
35      """ Main Program """
36      pygame.init()
37
38      # Set the height and width of the screen
39      size = [constants.SCREEN_WIDTH, constants.SCREEN_HEIGHT]
40      screen = pygame.display.set_mode(size)
41
42      pygame.display.set_caption("Platformer with sprite sheets")
43
44      # Create the player
45      player = Player()
46
47      # Create all the levels
48      level_list = []
49      level_list.append(levels.Level_01(player))
50      level_list.append(levels.Level_02(player))
51
52      # Set the current level
53      current_level_no = 0
54      current_level = level
55
56      active_sprite_list = pygame.sprite.Group()
57      player.level = current_level
58
59      player.rect.x = 340
60      player.rect.y = constants.SCREEN_HEIGHT - player.rect.height
61      active_sprite_list.add(player)
62
63      #Loop until the user clicks the close button.
64      done = False
65
66      # Used to manage how fast the screen updates
67      clock = pygame.time.Clock()
```

Cria o grupo de sprites e posiciona Player

Faz a movimentação do Player

```
69      # -------- Main Program Loop -----------
70      while not done:
71          for event in pygame.event.get(): # User did something
72              if event.type == pygame.QUIT: # If user clicked close
73                  do                                          .s loop
74
75              if event.type == pygame.KEYDOWN:
76                  if event.key == pygame.K_LEFT:
77                      player.go_left()
78                  if event.key == pygame.K_RIGHT:
79                      player.go_right()
80                  if event.key == pygame.K_UP:
81                      player.jump()
82
83              if event.type == pygame.KEYUP:
84                  if event.key == pygame.K_LEFT and player.change_x < 0:
85                      player.stop()
86                  if event.key == pygame.K_RIGHT and player.change_x > 0:
87                      player.stop()
88
89          # Update the player.
90          active_sprite_list.update()
91
92          # Update items in the level
93          current_level.update()
94
95          # If the player gets near the right side, shift the world left (-x)
96          if player.rect.x >= 500:
97              diff = player.rect.x - 500
98              player.rect.x = 500
99              current_level.shift_world(-diff)
```

```
69      # -------- Main Program Loop -----------
70      while not done:
71          for event in pygame.event.get(): # User did something
72              if event.type == pygame.QUIT: # If user clicked close
73                  done = True # Flag that we are done so we exit this loop
74
75              if event.type == pygame.KEYDOWN:
76                  if event.key == pygame.K_LEFT:
77                      player.go_left()
78                  if event.key == pygame.K_RIGHT:
79                      player.go_right()
80                  if event.key == pygame.K_UP:
81                      player.jump()
82
83              if event.type == pygame.KEYUP:
84                  if event.key == pygame.K_LEFT and player.change_x < 0:
85                      player.stop()
86                  if event.key == pygame.K_RIGHT and player.change_x > 0:
87                      player.stop()
88
89          # Update the player.
90          active_sprite_list.update()
91
92          # Update items in the level
93          current_level.update()
94
95          # If the player gets near the right side, shift the world left (-x)
96          if player.rect.x >= 500:
97              diff = player.rect.x - 500
98              player.rect.x = 500
99              current_level.shift_world(-diff)
```

Evita o Player sair da tela

platform_scroller.py

```
    # -------- Main Program Loop -----------
    while not done:
        for event in pygame.event.get(): # User did something
            if event.type == pygame.QUIT: # If user clicked close
                done = True # Flag that we are done so we exit this loop

            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_LEFT:
                    player.go_left()
                if event.key == pygame.K_RIGHT:
                    player.go_right()
                if event.key == pygame.K_UP:
                    player.jump()

            if event.type == pygame.KEYUP:
                if event.key == pygame.K_LEFT and player.change_x < 0:
                    player.stop()
                if event.key == pygame.K_RIGHT and player.change_x > 0:
                    player.stop()

        # Update the player.
        active_sprite_list.update()

        # Update items in the level
        current_level.update()

        # If the player gets near the right side, shift the world left (-x)
        if player.rect.x >= 500:
            diff = player.rect.x - 500
            player.rect.x = 500
            current_level.shift_world(-diff)
```

Atualiza player e níveis

platform_scroller.py

```
69      # -------- Main Program Loop -----------
70      while not done:
71          for event in pygame.event.get(): # User did something
72              if event.type == pygame.QUIT: # If user clicked close
73                  done = True # Flag that we are done so we exit this loop
74
75              if event.type == pygame.KEYDOWN:
76                  if event.key == pygame.K_LEFT:
77                      player.go_left()
78                  if event.key == pygame.K_RIGHT:
79                      player.go_right()
80                  if event.key == pygame.K_UP:
81                      player.jump()
82
83              if event.type == pygame.KEYUP:
84                  if event.key == pygame.K_LEFT and player.change_x < 0:
85                      player.stop()
86                  if event.key == pygame.K_RIGHT and player.change_x > 0:
87                      player.stop()
88
89          # Update the player.
90          active_sprite_list.update()
91
92
93
94
95          # If the player gets near the right side, shift the world left (-x)
96          if player.rect.x >= 500:
97              diff = player.rect.x - 500
98              player.rect.x = 500
99              current_level.shift_world(-diff)
```

Movimenta a tela do jogo quando o Player chega no extremo direito.

```
        # If the player gets near the left side, shift the world right (+x)
        if player.rect.x <= 120:
            diff = 120 - player.rect.x
            player.rect.x = 120
            current_level.shift_world(diff)
```

Movimenta a tela do jogo quando o Player chega no extremo esquerdo.

```
        if current_position < current_level.level_limit:
            player.rect.x = 120
            if current_level_no < len(level_list)-1:
                current_level_no += 1
                current_level = level_list[current_level_no]
                player.level = current_level

        # ALL CODE TO DRAW SHOULD GO BELOW THIS COMMENT
        current_level.draw(screen)
        active_sprite_list.draw(screen)

        # ALL CODE TO DRAW SHOULD GO ABOVE THIS COMMENT

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

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

    # Be IDLE friendly. If you forget this line, the program will 'hang'
    # on exit.
    pygame.quit()

if __name__ == "__main__":
    main()
```

platform_scroller.py

```
101         # If the player gets near the left side, shift the world right (+x)
102         if player.rect.x <= 120:
103             diff = 120 - player.rect.x
104
105
106
107         # If the player gets to the end of the level, go to the next level
108         current_position = player.rect.x + current_level.world_shift
109         if current_position < current_level.level_limit:
110             player.rect.x = 120
111             if current_level_no < len(level_list)-1:
112                 current_level_no += 1
113                 current_level = level_list[current_level_no]
114                 player.level = current_level
115
116         # ALL CODE TO DRAW SHOULD GO BELOW THIS COMMENT
117         current_level.draw(screen)
118         active_sprite_list.draw(screen)
119
120         # ALL CODE TO DRAW SHOULD GO ABOVE THIS COMMENT
121
122         # Limit to 60 frames per second
123         clock.tick(60)
124
125         # Go ahead and update the screen with what we've drawn.
126         pygame.display.flip()
127
128     # Be IDLE friendly. If you forget this line, the program will 'hang'
129     # on exit.
130     pygame.quit()
131
132 if __name__ == "__main__":
133     main()
```

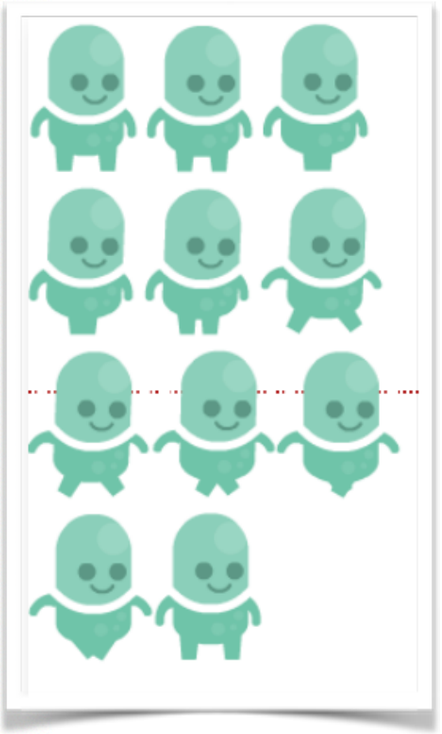
Testa se o Player chegou no final do primeiro nível (level)

platform_scroller.py

player.py

```
31
32      # -- Methods
33      def __init__(self):
34          """ Constructor function """
35
36          # Call the parent's constructor
37          pygame.sprite.Sprite.__init__(self)
38
39          sprite_sheet = SpriteSheet("p1_walk.png")
40          # Load all the right facing images into a list
41          image = sprite_sheet.get_image(0, 0, 66, 90)
42          self.walking_frames_r.append(image)
43          image = sprite_sheet.get_image(66, 0, 66, 90)
44          self.walking_frames_r.append(image)
45          image = sprite_sheet.get_image(132, 0, 67, 90)
46          self.walking_frames_r.append(image)
47          image = sprite_sheet.get_image(0, 93, 66, 90)
48          self.walking_frames_r.append(image)
49          image = sprite_sheet.get_image(66, 93, 66, 90)
50          self.walking_frames_r.append(image)
51          image = sprite_sheet.get_image(132, 93, 72, 90)
52          self.walking_frames_r.append(image)
53          image = sprite_sheet.get_image(0, 186, 70, 90)
54          self.walking_frames_r.append(image)
55
56          # Load all the right facing images, then flip them
57          # to face left.
58          image = sprite_sheet.get_image(0, 0, 66, 90)
59          image = pygame.transform.flip(image, True, False)
60          self.walking_frames_l.append(image)
61          image = sprite_sheet.get_image(66, 0, 66, 90)
62          image = pygame.transform.flip(image, True, False)
63          self.walking_frames_l.append(image)
```

p1_walk.png



player.py

platforms.py

```
"""
Module for managing platforms.
"""
import pygame

from spritesheet_functions import SpriteSheet

# These constants define our platform types:
#   Name of file
#   X location of sprite
#   Y location of sprite
#   Width of sprite
#   Height of sprite

GRASS_LEFT              = (576, 720, 70, 70)
GRASS_RIGHT             = (576, 576, 70, 70)
GRASS_MIDDLE            = (504, 576, 70, 70)
STONE_PLATFORM_LEFT     = (432, 720, 70, 40)
STONE_PLATFORM_MIDDLE = (648, 648, 70, 40)
STONE_PLATFORM_RIGHT  = (792, 648, 70, 40)

class Platform(pygame.sprite.Sprite):
    """ Platform the user can jump on """

    def __init__(self, sprite_sheet_data):
        """ Platform constructor. Assumes constructed with user passing in
            an array of 5 numbers like what's defined at the top of this
            code. """
        pygame.sprite.Sprite.__init__(self)

        sprite_sheet = SpriteSheet("tiles_spritesheet.png")
        # Grab the image for this platform
        self.image = sprite_sheet.get_image(sprite_sheet_data[0],
                                            sprite_sheet_data[1],
                                            sprite_sheet_data[2],
                                            sprite_sheet_data[3])
```

levels.py

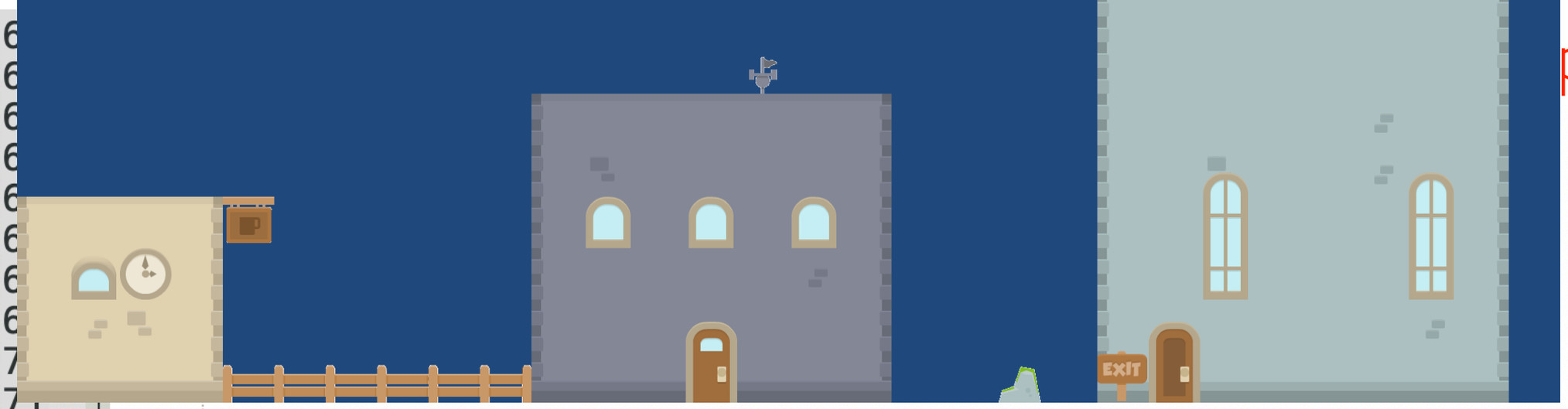
```
 6  class Level():
 7      """ This is a generic super-class used to define a level.
 8          Create a child class for each level with level-specific
 9          info. """
10
11      # Lists of sprites used in all levels. Add or remove
12      # lists as needed for your game. """
13      platform_list = None
14      enemy_list = None
15
16      # Background image
17      background = None
18
19      # How far this world has been scrolled left/right
20      world_shift = 0
21      level_limit = -1000
22
23      def __init__(self, player):
24          """ Constructor. Pass in a handle to player. Needed for when moving p
25              collide with the player. """
26          self.platform_list = pygame.sprite.Group()
27          self.enemy_list = pygame.sprite.Group()
28          self.player = player
29
30      # Update everythign on this level
31      def update(self):
32          """ Update everything in this level."""
33          self.platform_list.update()
34          self.enemy_list.update()
```

levels.py

```
36      def draw(self, screen):
37          """ Draw everything on this level. """
38
39          # Draw the background
40          # We don't shift the background as much as the sprites are shifted
41          # to give a feeling of depth.
42          screen.fill(constants.BLUE)
43          screen.blit(self.background,(self.world_shift // 3,0))
44
45          # Draw all the sprite lists that we have
46          self.platform_list.draw(screen)
47          self.enemy_list.draw(screen)
48
49      def shift_world(self, shift_x):
50          """ When the user moves left/right and we need to scroll everything:
51
52          # Keep track of the shift amount
53          self.world_shift += shift_x
54
55          # Go through all the sprite lists and shift
56          for platform in self.platform_list:
57              platform.rect.x += shift_x
58
59          for enemy in self.enemy_list:
60              enemy.rect.x += shift_x
```

levels.py

```
# Create platforms for the level
class Level_01(Level):
    """ Definition for level 1. """

    def __init__(self, player):
        """ Create level 1. """

        # Call the parent constructor
        Level.__init__(self, player)

        self.background = pygame.image.load("background_01.png").convert()
        self.background.set_colorkey(constants.WHITE)
        self.level_limit = -2500

        # Array with type of platform, and x, y location of the platform.
        level = [ [platforms.GRASS_LEFT, 500, 500],
                  [platforms.GRASS_MIDDLE, 570, 500],
                  [platforms.GRASS_RIGHT, 640, 500],
                  [platforms.GRASS_LEFT, 800, 400],
                  [platforms.GRASS_MIDDLE, 870, 400],
                  [platforms.GRASS_RIGHT, 940, 400],
                  [platforms.GRASS_LEFT, 1000, 500],
                  [platforms.GRASS_MIDDLE, 1070, 500],
                  [platforms.GRASS_RIGHT, 1140, 500],
                  [platforms.STONE_PLATFORM_LEFT, 1120, 280],
                  [platforms.STONE_PLATFORM_MIDDLE, 1190, 280],
                  [platforms.STONE_PLATFORM_RIGHT, 1260, 280],
                ]

        # Go through the array above and add platforms
        for platform in level:
            block = platforms.Platform(platform[0])
            block.rect.x = platform[1]
            block.rect.y = platform[2]
            block.player = self.player
            self.platform_list.add(block)
```



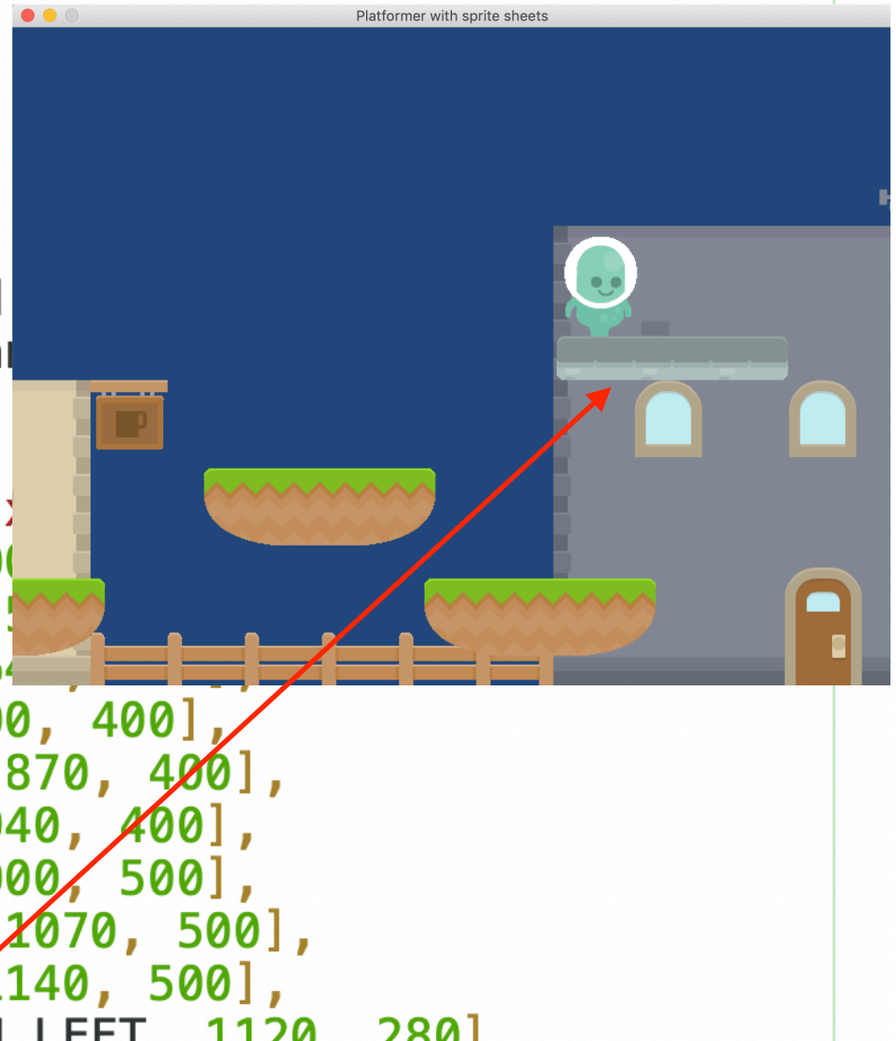

```
72          self.background = pygame.image.load("background_01.png").convert()
73          self.background.set_colorkey(constants.WHITE)
74          self.level_limit = -2500
75
76          # Array with type of platform, and x, y location of the platform.
77          level = [ [platforms.GRASS_LEFT, 500, 500],
78                    [platforms.GRASS_MIDDLE, 570, 500],
79                    [platforms.GRASS_RIGHT, 640, 500],
80                    [platforms.GRASS_LEFT, 800, 400],
81                    [platforms.GRASS_MIDDLE, 870, 400],
82                    [platforms.GRASS_RIGHT, 940, 400],
83                    [platforms.GRASS_LEFT, 1000, 500],
84                    [platforms.GRASS_MIDDLE, 1070, 500],
85                    [platforms.GRASS_RIGHT, 1140, 500],
86                    [platforms.STONE_PLATFORM_LEFT, 1120, 280],
87                    [platforms.STONE_PLATFORM_MIDDLE, 1190, 280],
88                    [platforms.STONE_PLATFORM_RIGHT, 1260, 280],
89                    ]
90
91          # Go through the array above and add platforms
92          for platform in level:
93              block = platforms.Platform(platform[0])
94              block.rect.x = platform[1]
95              block.rect.y = platform[2]
96              block.player = self.player
97              self.platform_list.add(block)
```

levels.py

```
# Create platforms for the level
class Level_01(Level):
    """ Definition for level 1. """

    def __init__(self, player):
        """ Create level 1. """

        # Call the parent constructor
        Level.__init__(self, player)

        self.background = pygame.image.load("background_01.png").convert()
        self.background.set_colorkey(constants.WHITE)
        self.level_limit = -2500

        # Array with type of platform, and x, y location of the platform.
        level = [ [platforms.GRASS_LEFT, 500, 500],
                  [platforms.GRASS_MIDDLE, 570, 500],
                  [platforms.GRASS_RIGHT, 640, 500],
                  [platforms.GRASS_LEFT, 800, 400],
                  [platforms.GRASS_MIDDLE, 870, 400],
                  [platforms.GRASS_RIGHT, 940, 400],
                  ms.GRASS_LEFT, 1000, 500],
                  ms.GRASS_MIDDLE, 1070, 500],
                  ms.GRASS_RIGHT, 1140, 500],
                  ms.STONE_PLATFORM_LEFT, 1120, 280],
                  ms.STONE_PLATFORM_MIDDLE, 1190, 280],
                  ms.STONE_PLATFORM_RIGHT, 1260, 280],


        rray above and add platforms
        vel:
        rms.Platform(platform[0])
         platform[1]
         platform[2]
         self.player
        list.add(block)
```

levels.py

```
# Create platforms for the level
class Level_01(Level):
    """ Definition for level 1. """

    def __init__(self, player):
        """ Create level 1. """

        # Call the parent constructor
        Level.__init__(self, player)

        self.background = pygame.image.load
        self.background.set_colorkey(constar
        self.level_limit = -2500

        # Array with type of platform, and x
        level = [ [platforms.GRASS_LEFT, 500
                  [platforms.GRASS_MIDDLE, 5
                  [platforms.GRASS_RIGHT, 64
                  [platforms.GRASS_LEFT, 800, 400],
                  [platforms.GRASS_MIDDLE, 870, 400],
                  [platforms.GRASS_RIGHT, 940, 400],
                  [platforms.GRASS_LEFT, 1000, 500],
                  [platforms.GRASS_MIDDLE, 1070, 500],
                  [platforms.GRASS_RIGHT, 1140, 500],
                  [platforms.STONE_PLATFORM_LEFT, 1120, 280],
                  [platforms.STONE_PLATFORM_MIDDLE, 1190, 280],
                  [platforms.STONE_PLATFORM_RIGHT, 1260, 280],
                  ]

        # Go through the array above and add platforms
        for platform in level:
            block = platforms.Platform(platform[0])
            block.rect.x = platform[1]
            block.rect.y = platform[2]
            block.player = self.player
            self.platform_list.add(block)
```



Exercícios

1. **Utilize o código do exemplo sobre Sprite Sheets e utilize duas teclas, uma para aumentar a velocidade do jogo e outra para diminuir.**

2. **Adicione uma opção para pausar o jogo (também deve ser tratado o evento a partir de uma tecla).**

3. **Adicione uma tecla para finalizar o jogo.**