PyGame Tutorial

Introduction to Programming (I) Final Project

Prof. Pai H. Chou and Min-Chun Hu
Computer Science, National Tsing Hua University

Outline

Introduction to game programming

Introduction to 2D game

PyGame

Map editor utilities

→ Tiled

→ PyTMX

Outline

Introduction to game programming

Introduction to 2D game

PyGame

Map editor utilities

→ Tiled

→ PyTMX

Game Programming

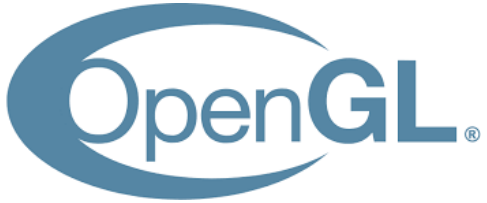
- **What is Game Programming?**
- **Designing** and **Coding** software that allows a game to function
- It involves creating gameplay mechanics, physics, AI, graphics, sound systems, etc.

Game Programming

Game Programming

Which tool to develop game?

UNREAL ENGINE

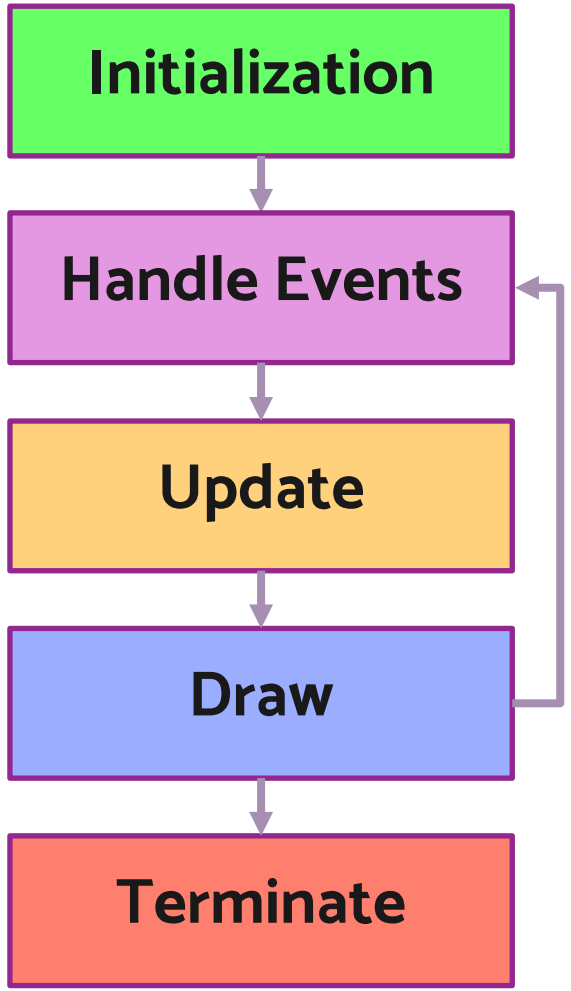


Low Level **High Level**

Game Programming

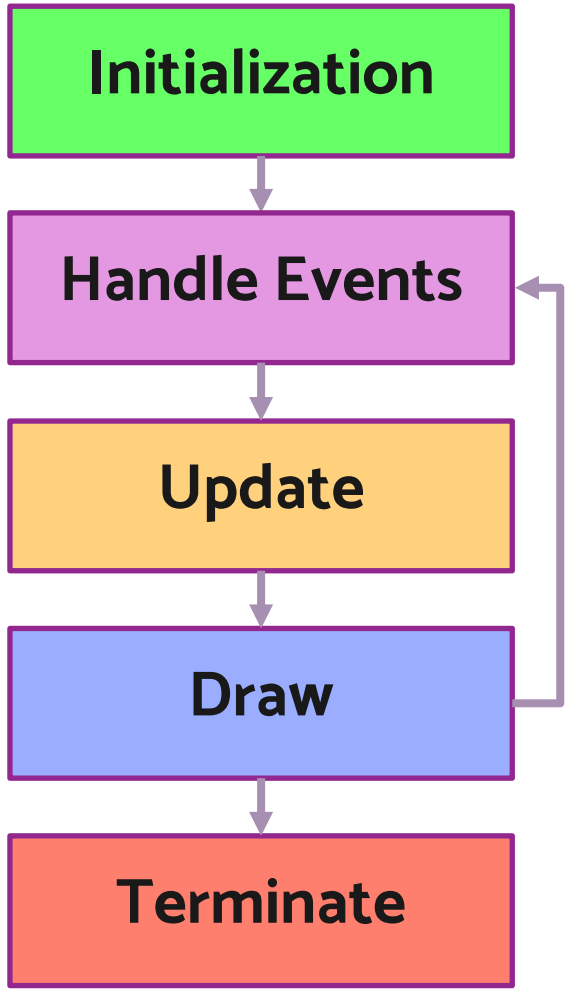
The pipeline of creating a game:

1. Initialization
2. Game loop
   a. Handle Events
   b. Update
   c. Draw
3. Terminate



Game Programming

- **Initialization** : Initialize your properties
- **Handle Events** : Handle events e.g. input
- **Update** : Game logic update
- **Draw** : Render your game
- **Terminate** : Cleanup space



Game Programming

Example in pseudocode

```
player_asset = load_image("assets/player.png")
player_position = (0, 0)
game_is_running = True
while(game_is_running):
    events = get_event()
    for ev in events:
        if ev == Quit:
            game_is_running = False
        if ev == Press_Key_W:
            player_position += (1, 0)
    draw(player_asset)
delete player_asset
```

Python

Outline

Introduction to game programming

Introduction to 2D game

PyGame

Map editor utilities

→ Tiled

→ PyTMX

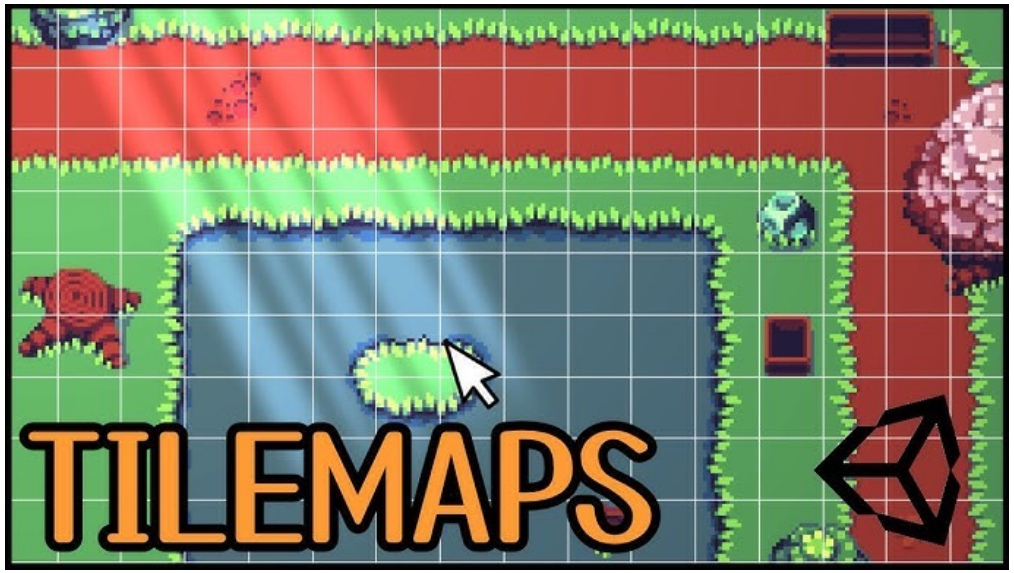
2D Game

2D Game

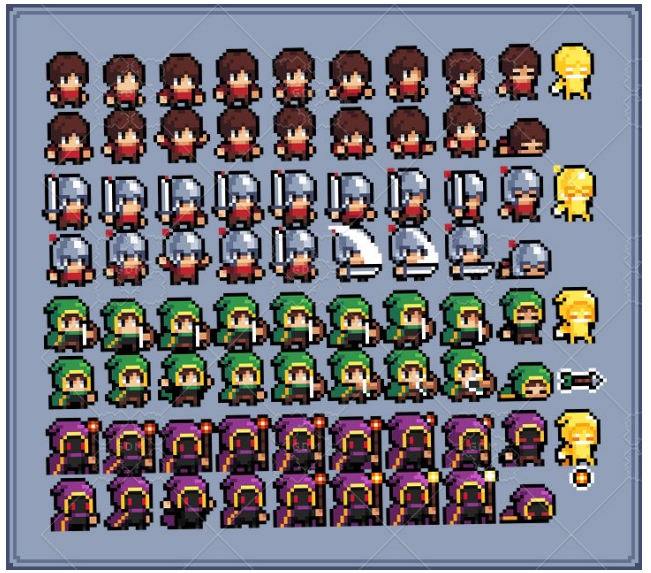
2D Assets Terminology:

- Sprite : 2D image or animation representing a character, object, etc.
- Tile : Small square image used to build maps or environments
- Tilemap : A grid of tiles that forms a full level or background
- Frame : A single image in an animation sequence
- Atlas : A single large image containing multiple sprites

2D Game

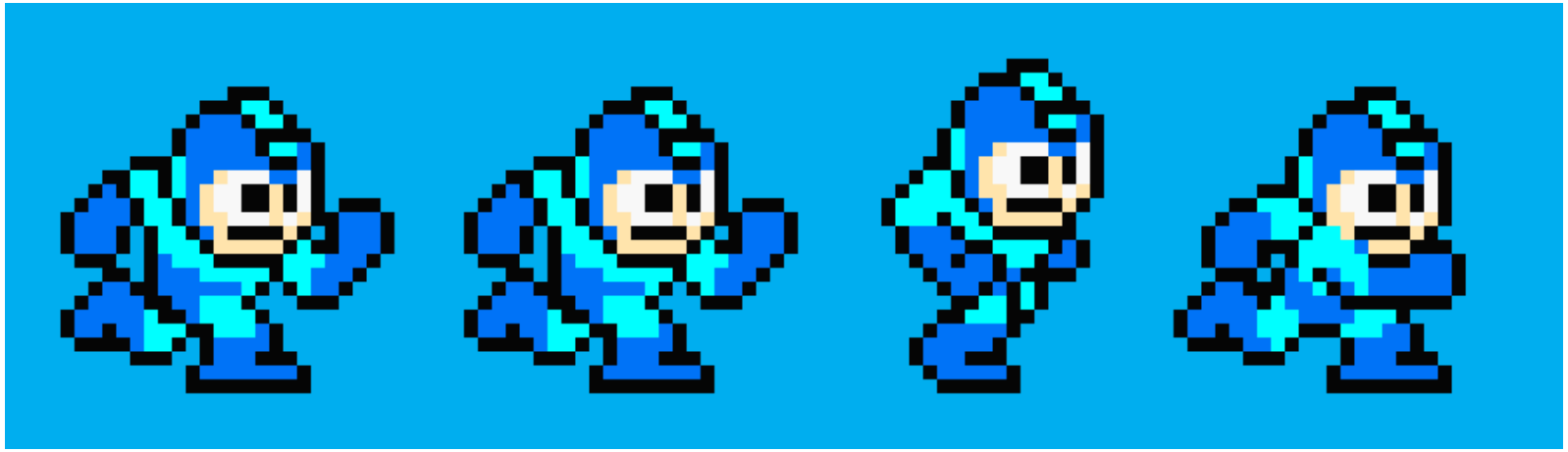


Tile & Tilemap



Atlas

2D Game



Animation = Frame 0 + Frame 1 + Frame 2

2D Game

2D Gameplay Terminology:

- Entity : Interactive object in the game world (player, enemy, …)
- Scene : One playable area or stage of the game

If there are more terminology you don’t understand, it’s recommended for you to google or to ask AI (e.g. ChatGPT)

Outline

Introduction to game programming

Introduction to 2D game

PyGame

Map editor utilities

→ Tiled

→ PyTMX

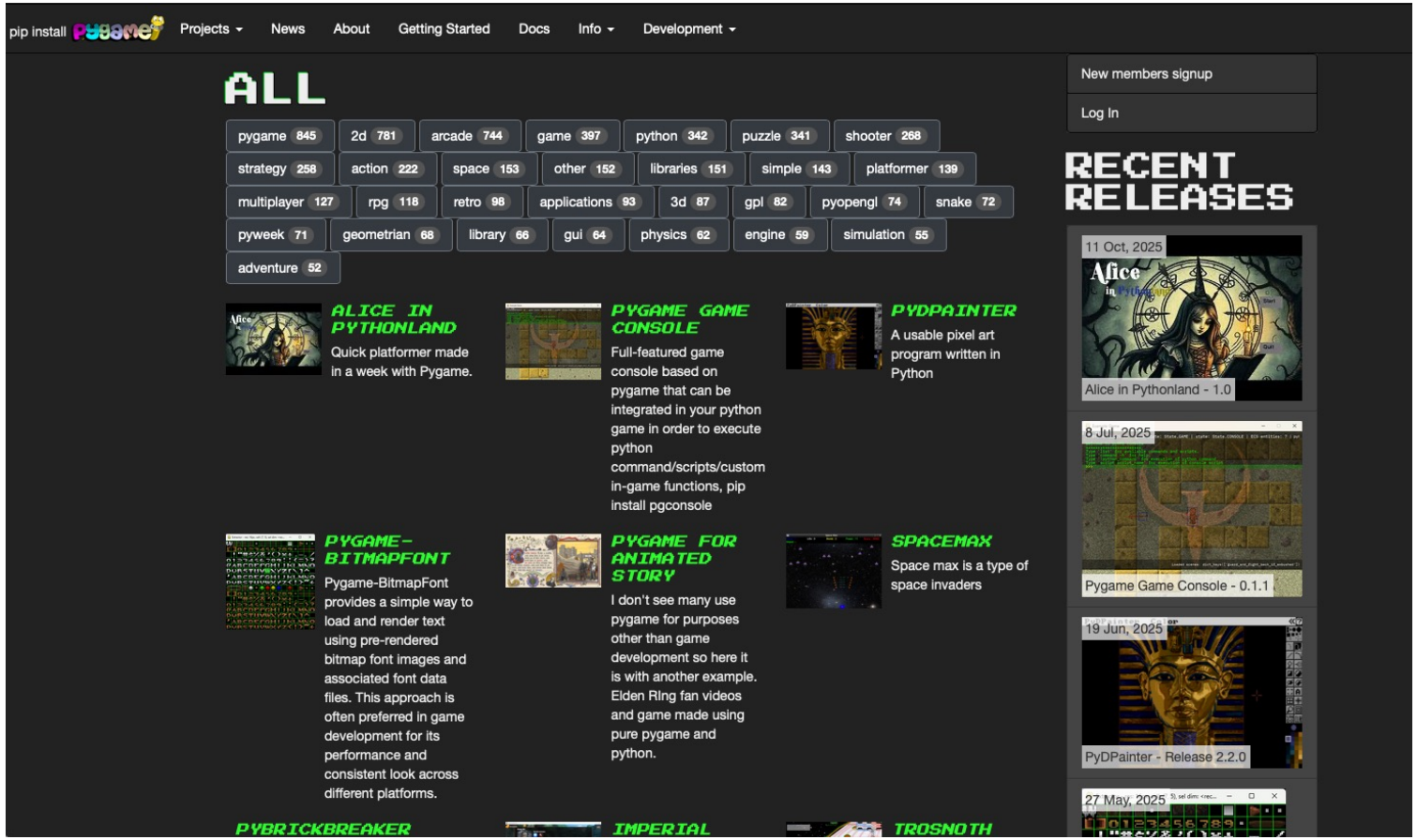
PyGame

PyGame : Python library to create a game

Official Website : https://www.pygame.org

Install PyGame : `pip install pygame`

PyGame


pip install pygame
Projects
News
About
Getting Started
Docs
Info
Development
ALL
pygame 845
2d 781
arcade 744
game 397
python 342
puzzle 341
shooter 268
strategy 258
action 222
space 153
other 152
libraries 151
simple 143
platformer 139
multiplayer 127
rpg 118
retro 98
applications 93
3d 87
gpl 82
pyopengl 74
snake 72
pyweek 71
geometrian 68
library 66
gui 64
physics 62
engine 59
simulation 55
adventure 52
ALICE IN PYTHONLAND
Quick platformer made in a week with Pygame.
PYGAME GAME CONSOLE
Full-featured game console based on pygame that can be integrated in your python game in order to execute python command/scripts/custom in-game functions, pip install pgconsole
PYDPAINTER
A usable pixel art program written in Python
PYGAME-BITMAPFONT
Pygame-BitmapFont provides a simple way to load and render text using pre-rendered bitmap font images and associated font data files. This approach is often preferred in game development for its performance and consistent look across different platforms.
PYGAME FOR ANIMATED STORY
I don't see many use pygame for purposes other than game development so here it is with another example. Elden RIng fan videos and game made using pure pygame and python.
SPACEMAX
Space max is a type of space invaders
PYBRICKBREAKER
IMPERIAL
TROSNOTH
New members signup
Log In
RECENT RELEASES
11 Oct, 2025
Alice in Pythonland - 1.0
8 Jul, 2025
Pygame Game Console - 0.1.1
19 Jun, 2025
PyDPainter - Release 2.2.0
27 May, 2025

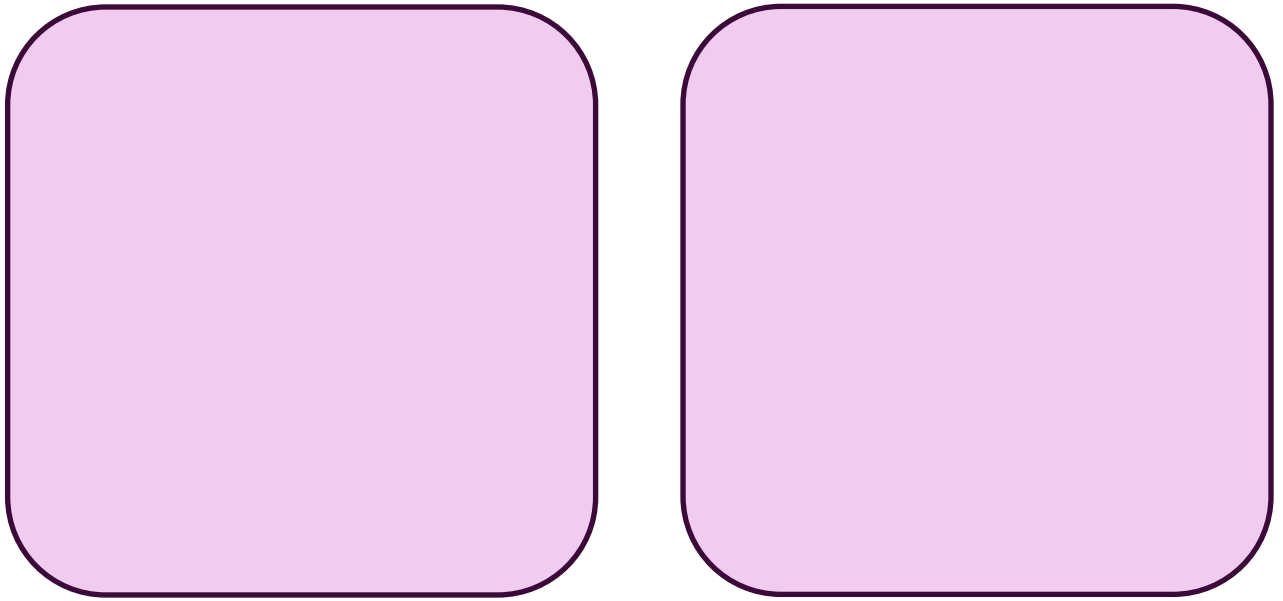
PyGame

Hello Window

```python
import pygame

pygame.init()

screen = pygame.display.set_mode((640, 480))
pygame.display.set_caption("Hello Window")

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

PyGame

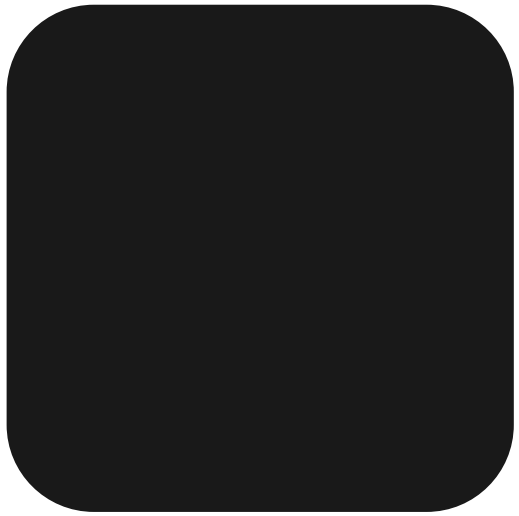
- Every program has 2 buffers
- **Front Buffer** – Image shown on current frame
- **Back Buffer** – The next frame being drawn off-screen



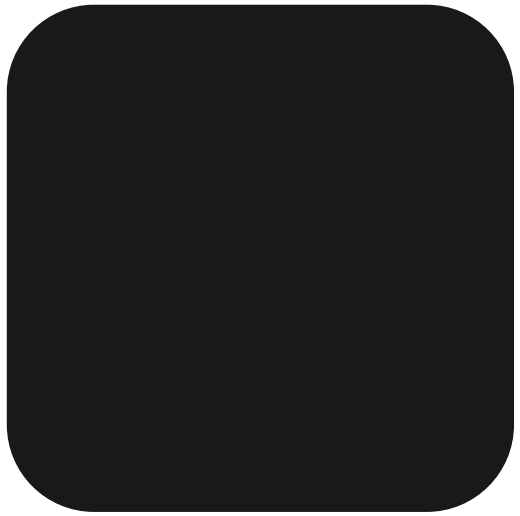
```python
import pygame

pygame.init()

screen = pygame.display.set_mode((640, 480))
pygame.display.set_caption("Hello Green")

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

    pygame.time.wait(10)
    screen.fill((0, 255, 0))
    pygame.display.flip()

    pygame.time.wait(10)
    screen.fill((255, 0, 0))
    pygame.display.flip()
```

Python

PyGame



Front Buffer

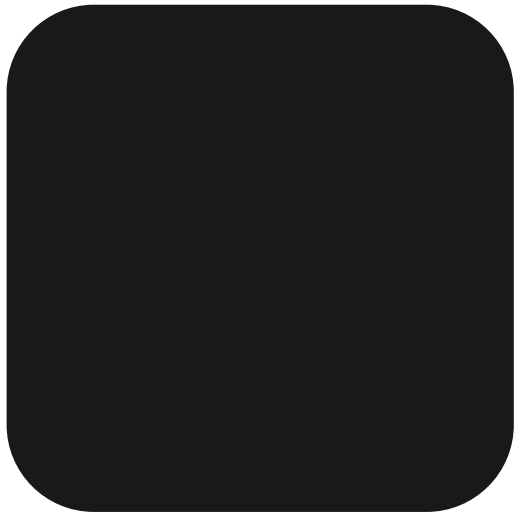


Back Buffer

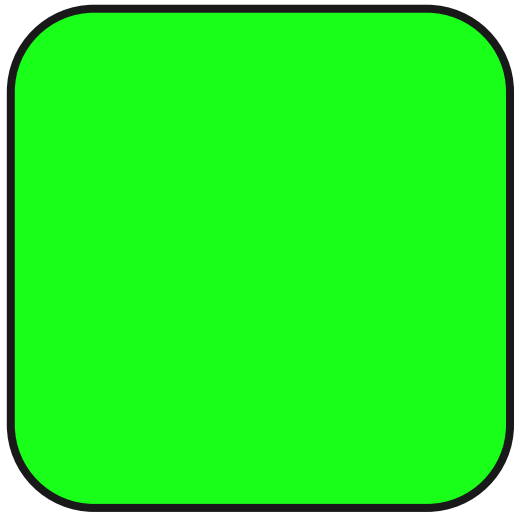
```python
import pygame

pygame.init()

screen = pygame.display.set_mode((640, 480))
pygame.display.set_caption("Hello Green")

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

    pygame.time.wait(10)
    screen.fill((0, 255, 0))
    pygame.display.flip()

    pygame.time.wait(10)
    screen.fill((255, 0, 0))
    pygame.display.flip()
```

Python

PyGame



Front Buffer

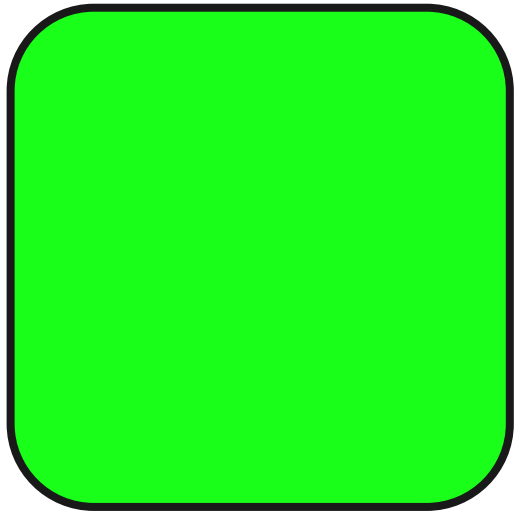


Back Buffer

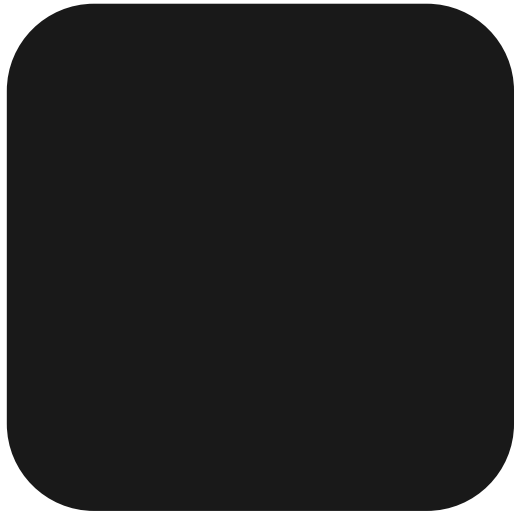
```python
import pygame

pygame.init()

screen = pygame.display.set_mode((640, 480))
pygame.display.set_caption("Hello Green")

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

    pygame.time.wait(10)
    screen.fill((0, 255, 0))
    pygame.display.flip()

    pygame.time.wait(10)
    screen.fill((255, 0, 0))
    pygame.display.flip()
```

Python

PyGame



Front Buffer

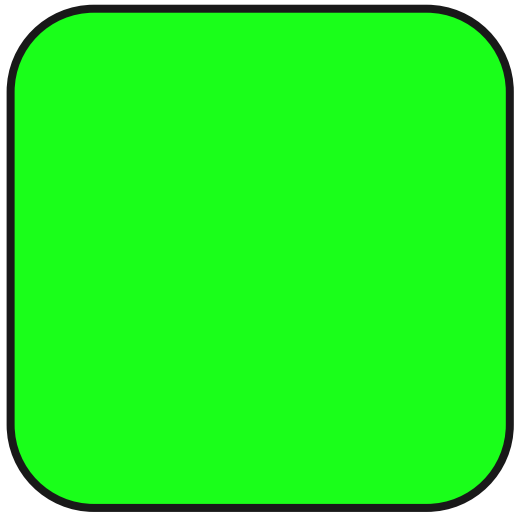


Back Buffer

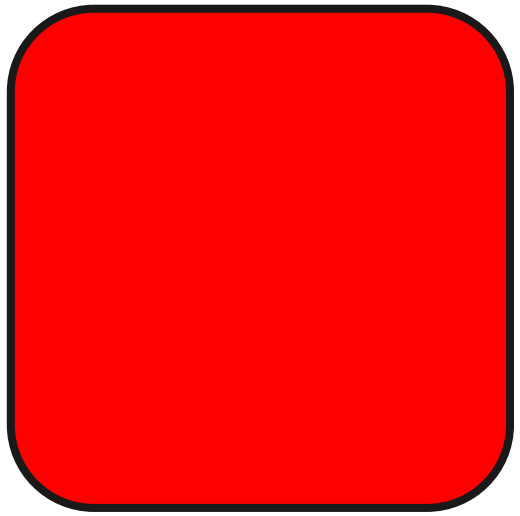
```python
import pygame

pygame.init()

screen = pygame.display.set_mode((640, 480))
pygame.display.set_caption("Hello Green")

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

    pygame.time.wait(10)
    screen.fill((0, 255, 0))
    pygame.display.flip()

    pygame.time.wait(10)
    screen.fill((255, 0, 0))
    pygame.display.flip()
```

Python

PyGame



Front Buffer

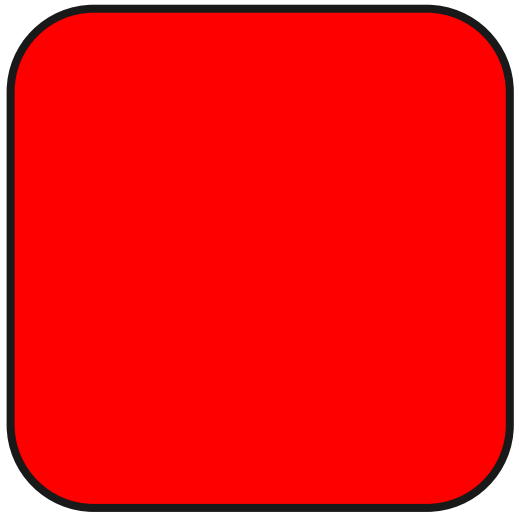


Back Buffer

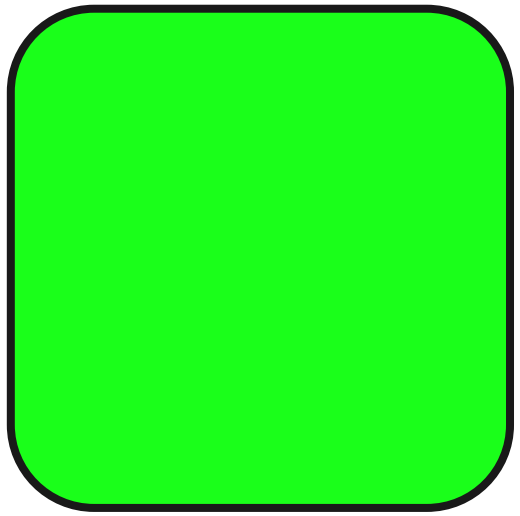
```python
import pygame

pygame.init()

screen = pygame.display.set_mode((640, 480))
pygame.display.set_caption("Hello Green")

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

    pygame.time.wait(10)
    screen.fill((0, 255, 0))
    pygame.display.flip()

    pygame.time.wait(10)
    screen.fill((255, 0, 0))
    pygame.display.flip()
```

Python

PyGame



Front Buffer



Back Buffer

PyGame

- Call `pygame.display.flip()` to update front buffer with back buffer
- As it will always send back buffer to front buffer, it's **undefined behaviour to what happen to the back buffer**, so it's better to redraw your scene in back buffer.

PyGame

3 Basic Assets:

- Image
- Sound
- Font

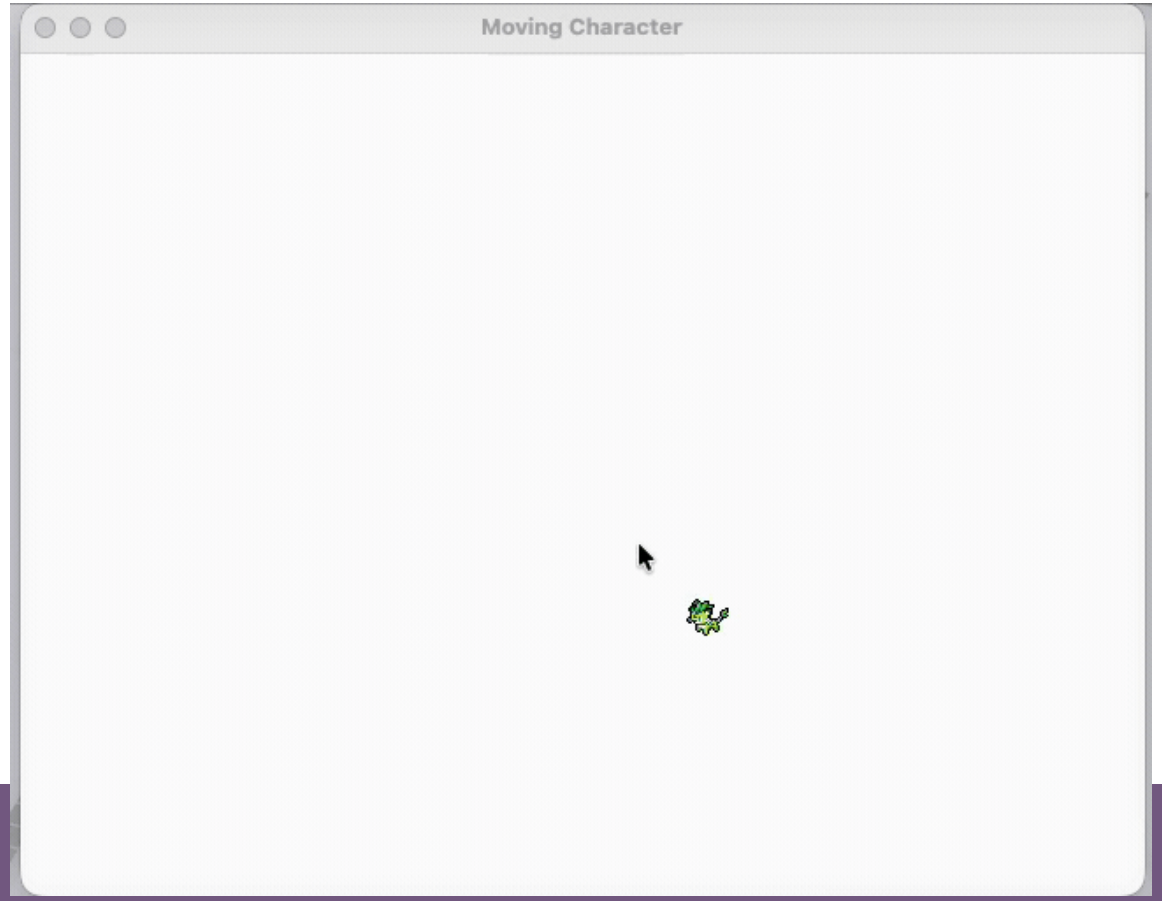
How to load and use image, sound, and font?

PyGame

```python
# Load image
player_img = pygame.image.load("player.png")
screen.blit(player_img, (100, 100))  # draw image

# Load audio
jump_sound = pygame.mixer.Sound("jump.wav")
jump_sound.play()  # play sound

# Load font
font = pygame.font.Font("arial.ttf", 32)
text = font.render("Hello!", True, (255, 255, 255))
screen.blit(text, (50, 50))  # draw text

pygame.display.flip()
```

PyGame

Moving Character



PyGame

```python
import pygame

pygame.init()

screen = pygame.display.set_mode((640, 480))
pygame.display.set_caption("Moving Character")

player = pygame.image.load('player.png')
player_position = (0, 0)
player_speed = 25
...
```

PyGame

```python
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                player_position = (player_position[0] - player_speed, player_position[1])
            elif event.key == pygame.K_RIGHT:
                player_position = (player_position[0] + player_speed, player_position[1])
            elif event.key == pygame.K_UP:
                player_position = (player_position[0], player_position[1] - player_speed)
            elif event.key == pygame.K_DOWN:
                player_position = (player_position[0], player_position[1] + player_speed)
    screen.fill((255, 255, 255))
    screen.blit(player, player_position)
    pygame.display.flip()
```

Python

Outline

Introduction to game programming

Introduction to 2D game

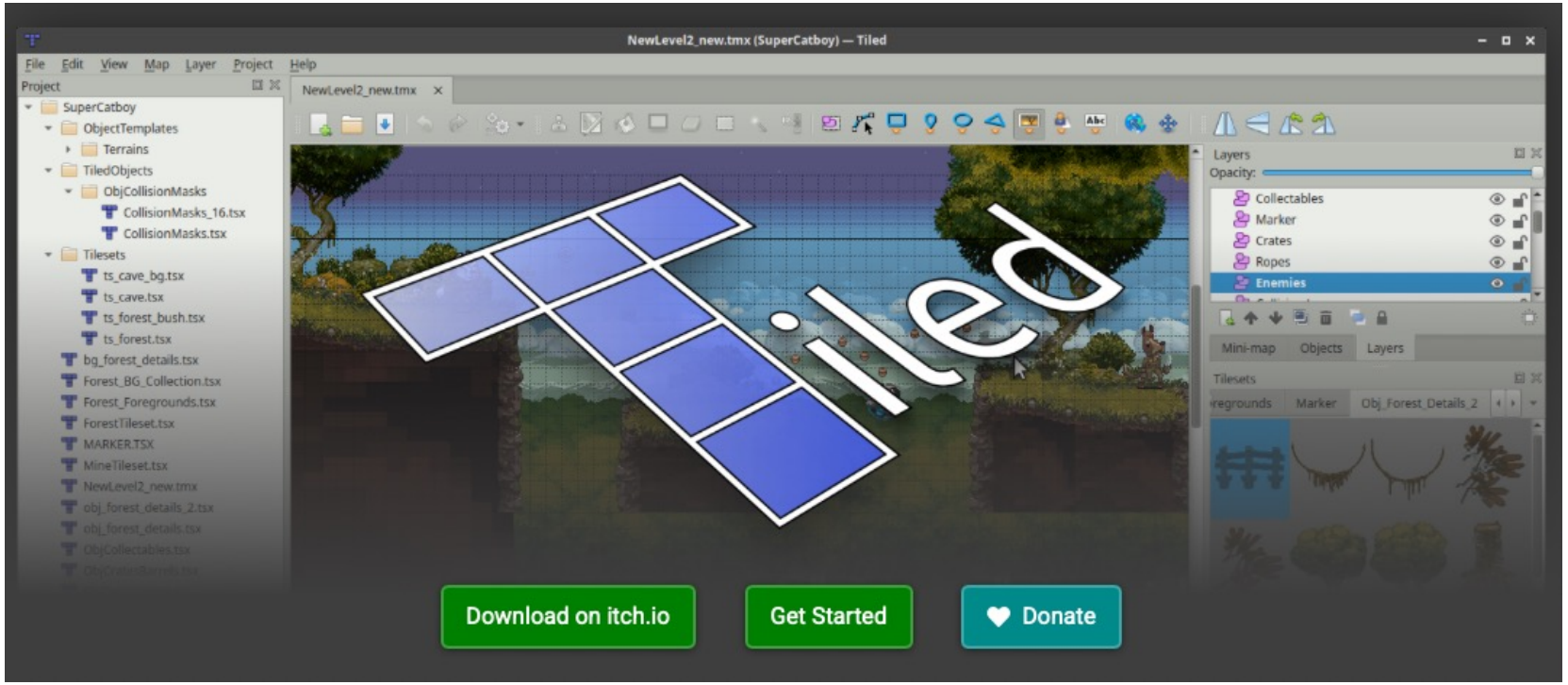
PyGame

Map editor utilities

→ Tiled

→ PyTMX

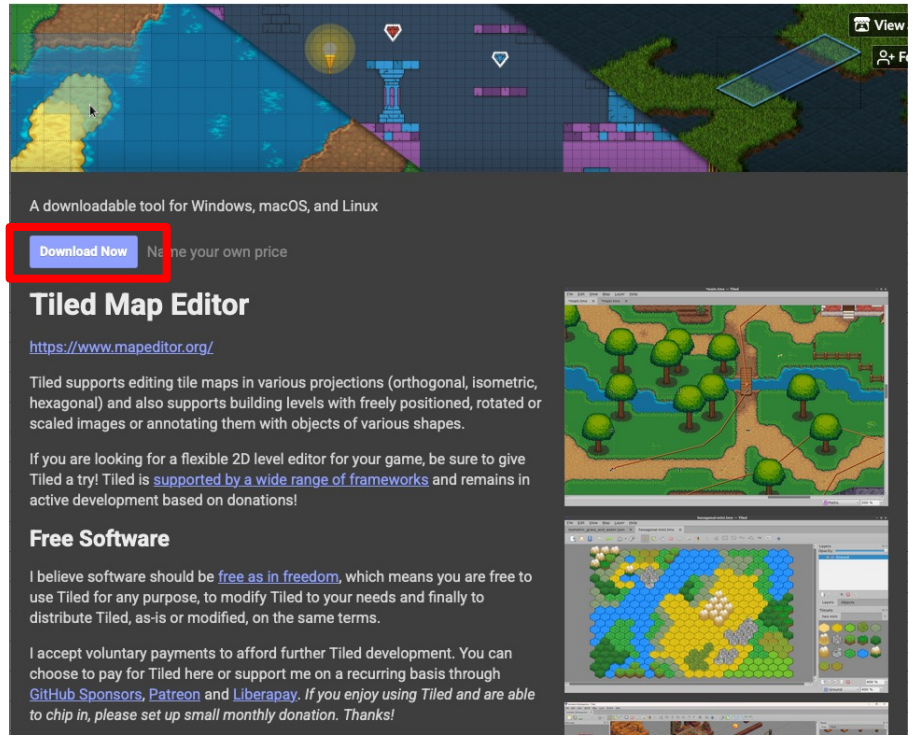
Tiled

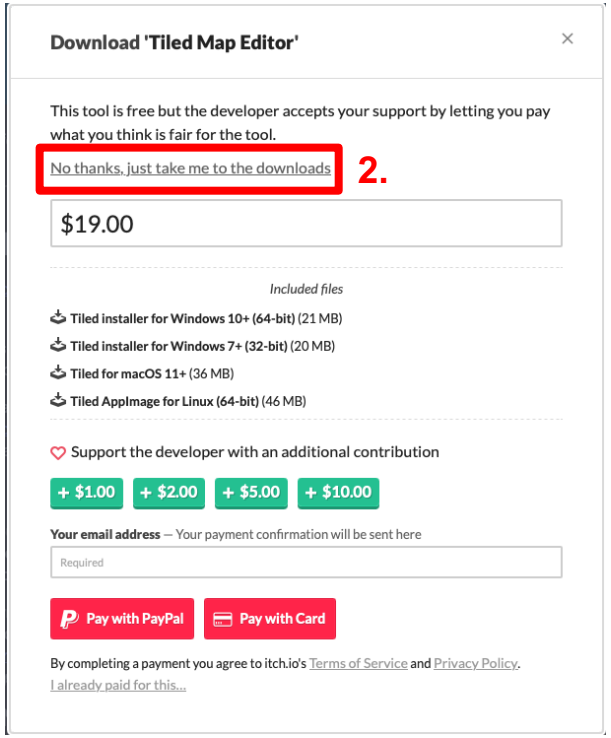


https://www.mapeditor.org

Tiled

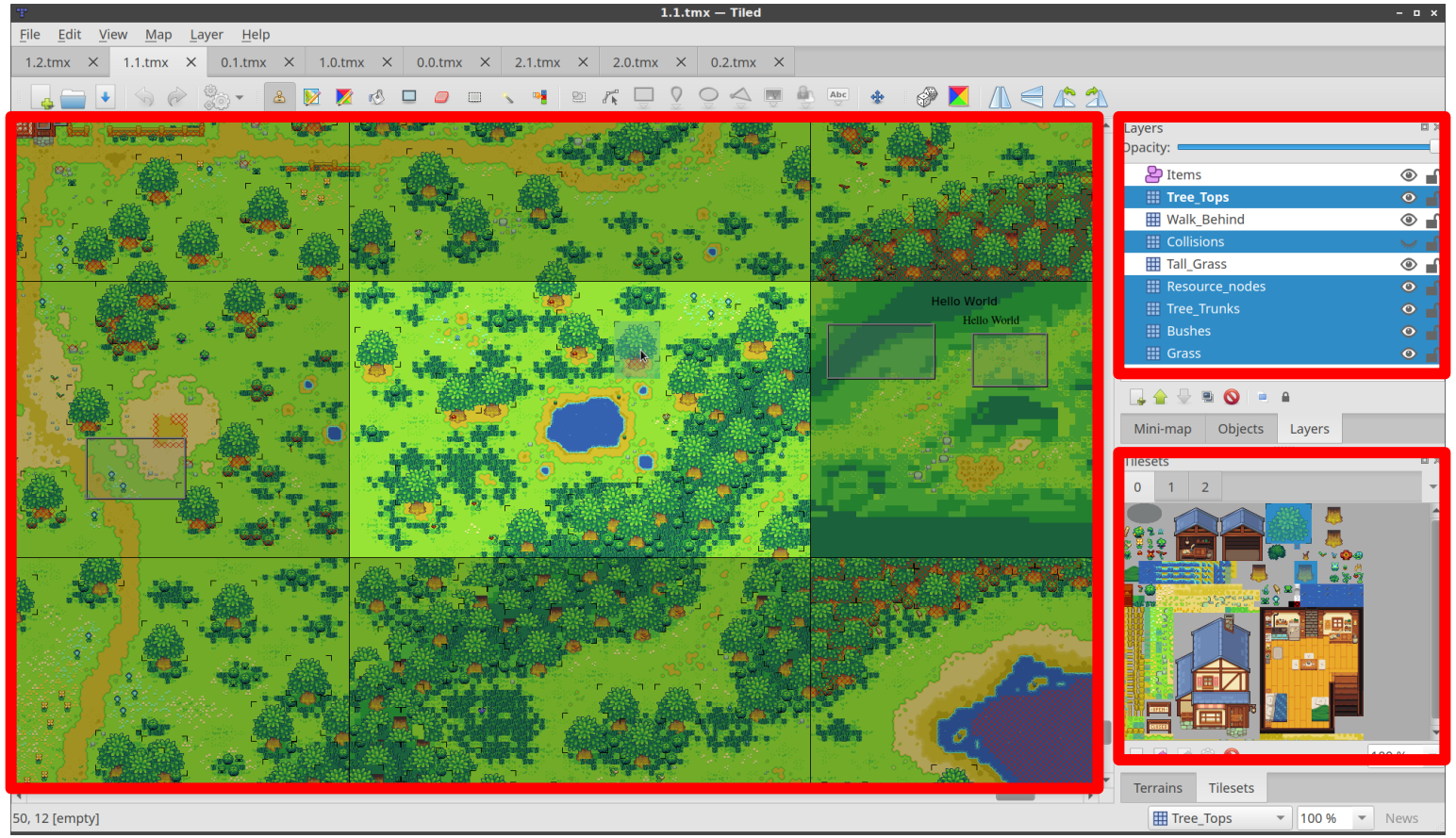
Download Tiled





Tiled

Your Map



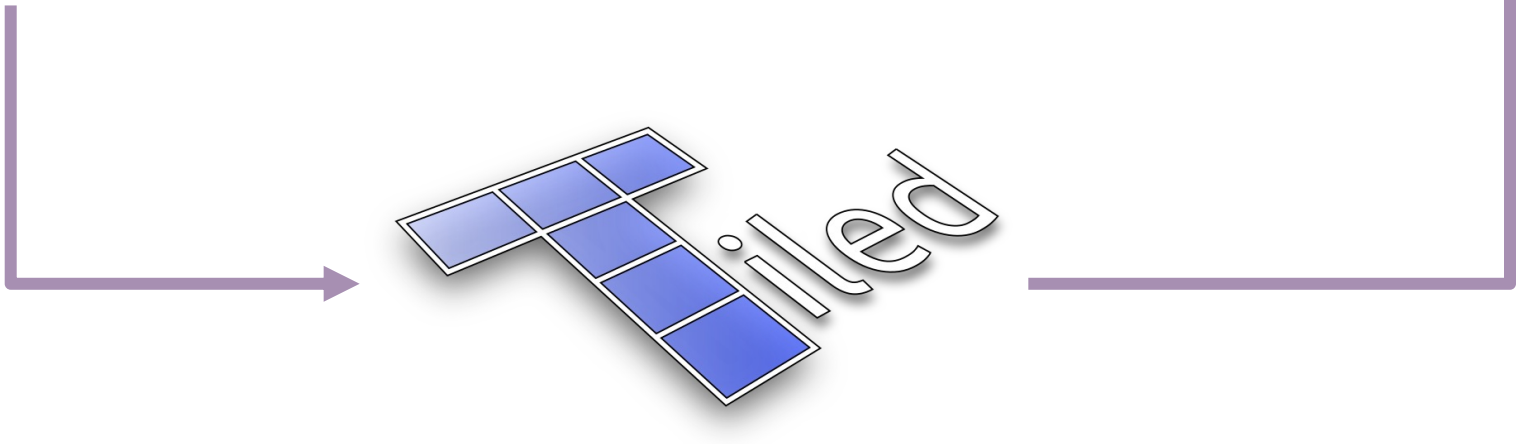
Layer

Tileset Atlas

Tiled

In this project, we will provide you this atlas, you can try to make your own map

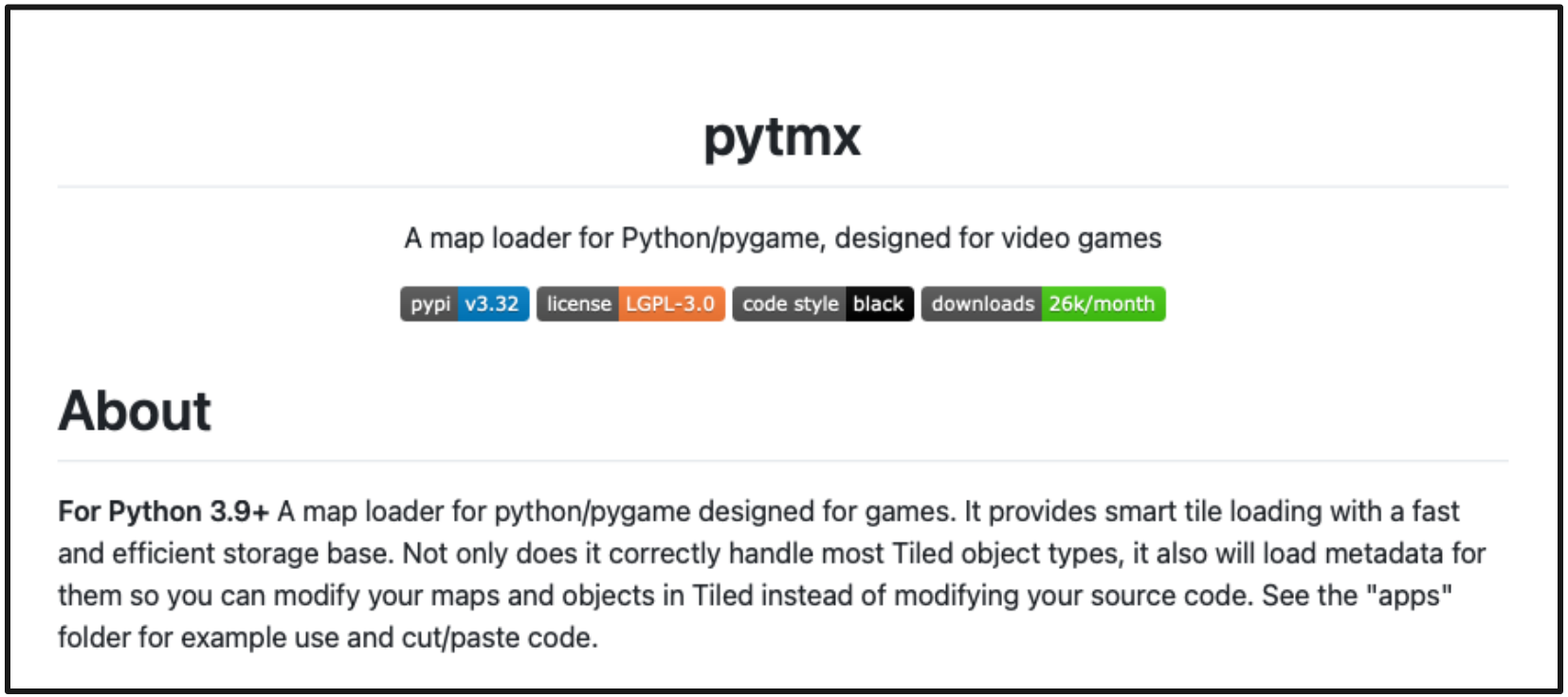
Tiled



Tiled

How do we parse it into Python?

PyTMX



pytmx

A map loader for Python/pygame, designed for video games

pypi v3.32 | license LGPL-3.0 | code style black | downloads 26k/month

About

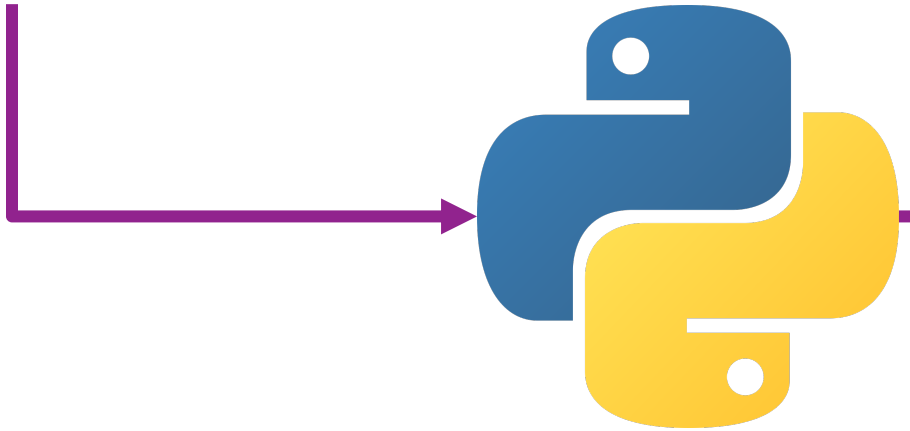
For Python 3.9+ A map loader for python/pygame designed for games. It provides smart tile loading with a fast and efficient storage base. Not only does it correctly handle most Tiled object types, it also will load metadata for them so you can modify your maps and objects in Tiled instead of modifying your source code. See the "apps" folder for example use and cut/paste code.

https://github.com/bitcraft/pytmx

PyTMX

map.tmx

tileset.tsx



+ PyTMX

Tiled & PyTMX

```python
import sys
import pygame as pg
from pytmx.util_pygame import load_pygame
from pytmx import TiledTileLayer


pg.init()
screen = pg.display.set_mode((640, 480))
pg.display.set_caption("Tilemap")


tmxdata = load_pygame("assets/maps/map.tmx")


# Create a canvas for images
tile_w, tile_h = tmxdata.tilewidth, tmxdata.tileheight
pixel_w, pixel_h = tmxdata.width * tile_w, tmxdata.height * tile_h
surface = pg.Surface((pixel_w, pixel_h), pg.SRCALPHA)
```

Python

Tiled & PyTMX

```python
# Build a map once
for layer in tmxdata.visible_layers:
    if isinstance(layer, TiledTileLayer):
        for x, y, gid in layer:
            if gid == 0:
                continue
            image = tmxdata.get_tile_image_by_gid(gid)
            if image is None:
                continue
            if image.get_size() != (tile_w, tile_h):
                image = pg.transform.scale(image, (tile_w, tile_h))
            surface.blit(image, (x * tile_w, y * tile_h))
```

Python

Tiled & PyTMX

```python
running = True
while running:
    for event in pg.event.get():
        if event.type == pg.QUIT:
            running = False

    screen.fill((0, 0, 0))
    screen.blit(surface, (0, 0))
    pg.display.flip()

pg.quit()
sys.exit()
```

Python

Questions?