Criando jogos 3D com Ursina

(ou quase isso)

Live de Python #236

Como temos que explicar todos os conceitos iniciais do 3D, talvez o jogo não fique lá aquelas coisas!

Aviso 1!

Se vocês gostarem, podemos fazer uma segunda live nos aprofundando nos modelos e fazendo um jogo sem ter que explicar a base de tudo!

Aviso 2!

Roteiro

1. Uma base sobre 3D

Alguns conceitos básicos (**BEM INTRODUTÓRIO**)

2. Ursina

Conhecendo a biblioteca e as entidades

3. Tornando dinâmico

Lindando com updates e inputs de teclado

4. Fazendo um pequeno jogo

Até a hora que estourarmos o tempo!

picpay.me/dunossauro

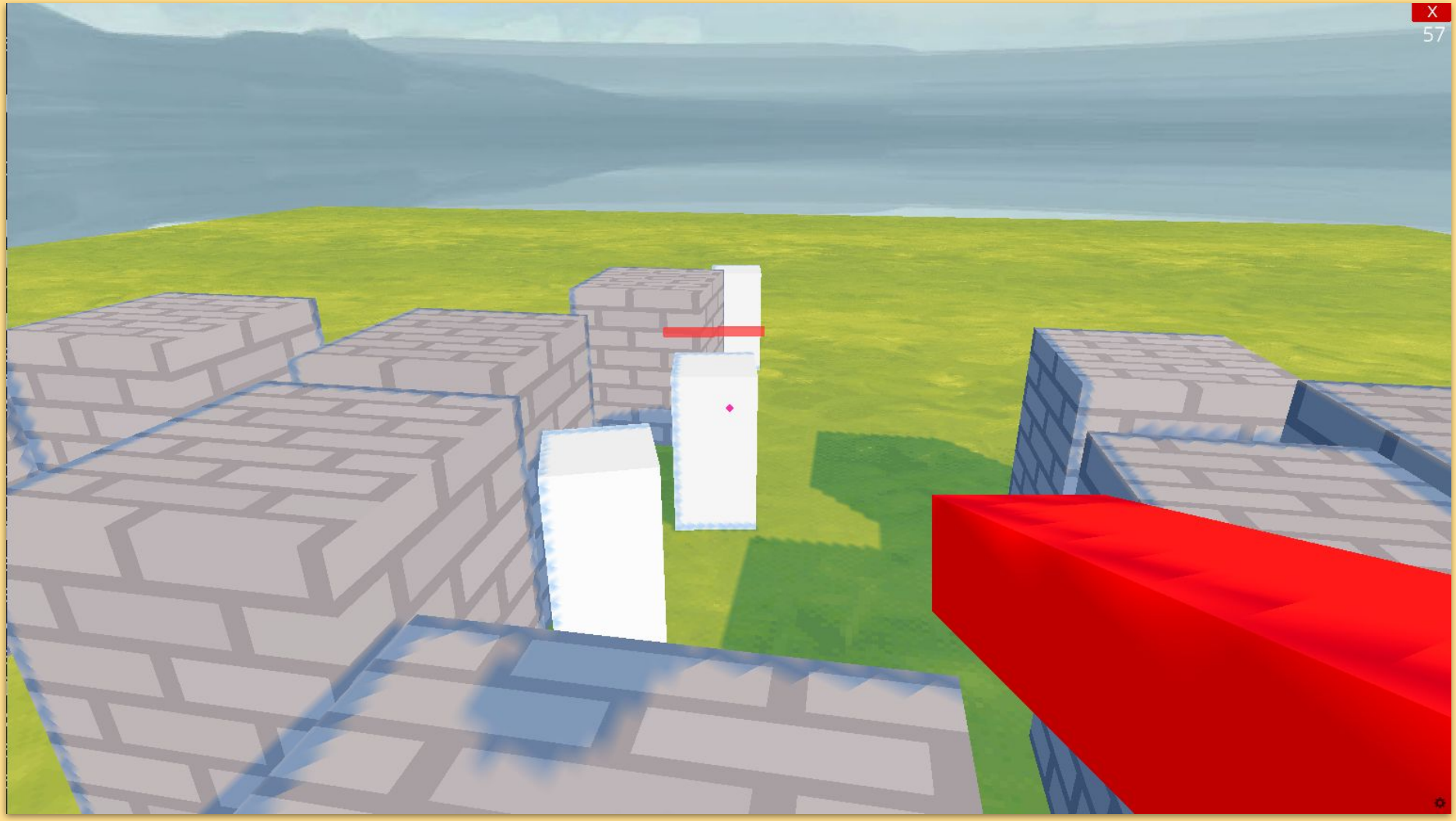
apoia.se/livedepython

pix.dunossauro@gmail.com

Ajude o projeto <3

Ademar Peixoto, Adilson Herculano, Adriano Ferraz, Alexandre Harano, Alexandre Lima, Alexandre Takahashi, Alexandre Villares, Alex Lima, Alynne Ferreira, Alysson Oliveira, Ana Carneiro, Andre Azevedo, Andre Mesquita, Aquiles Coutinho, Arnaldo Turque, Aslay Clevisson, Aurelio Costa, Bernardo At, Bernardo Fontes, Bruno Almeida, Bruno Barcellos, Bruno Barros, Bruno Batista, Bruno Freitas, Bruno Lopes, Bruno Ramos, Caio Nascimento, Christiano Morais, Damianth, Daniel Freitas, Daniel Wojcickoski, Danilo Boas, Danilo Segura, Danilo Silva, David Couto, David Kwast, Davi Goivinho, Delton Porfiro, Denis Bernardo, Dgeison Peixoto, Diego Farias, Diego Guimarães, Dilenon Delfino, Diogo Paschoal, Edgar, Edinilson Bonato, Eduardo Tolmasquim, Elias Silva, Emerson Rafael, Érico Andrei, Everton Silva, Fabiano, Fabiano Tomita, Fabio Barros, Fábio Barros, Fabio Castro, Fábio Thomaz, Felipe Rodrigues, Fernanda Prado, Fernando Celmer, Firehouse, Flávio Meira, Francisco Neto, Gabriel Barbosa, Gabriel Espindola, Gabriel Mizuno, Gabriel Moreira, Gabriel Paiva, Gabriel Simonetto, Gabriel Souza, Geandreson Costa, Gilberto Abrao, Giovanna Teodoro, Giuliano Silva, Guilherme Felitti, Guilherme Gall, Guilherme Silva, Guionardo Furlan, Gustavo Pereira, Gustavo Suto, Harold Gautschi, Heitor Fernandes, Helvio Rezende, Hugo Cosme, Igor Riegel, Italo Silva, Janael Pinheiro, Jhonatan Martins, Joelson Sartori, Jônatas Silva, Jon Cardoso, Jorge Silva, Jose Alves, José Gomes, Joseíto Júnior, Jose Mazolini, Juan Felipe, Juan Gutierrez, Juliana Machado, Julio Franco, Júlio Gazeta, Julio Silva, Kaio Peixoto, Kaneson Alves, Leandro Miranda, Leandro Silva, Leo Ivan, Leonardo Mello, Leonardo Nazareth, Leon Solon, Luancomputacao Roger, Lucas Adorno, Lucas Carderelli, Lucas Mendes, Lucas Nascimento, Lucas Schneider, Lucas Simon, Lucas Valino, Luciano Filho, Luciano Ratamero, Luciano Silva, Luciano Teixeira, Luis Alves, Luiz Duarte, Luiz Lima, Luiz Paula, Luiz Perciliano, Mackilem Laan, Marcelo Campos, Marcio Moises, Marco Mello, Marcos Gomes, Maria Clara, Marina Passos, Mateus Lisboa, Matheus Silva, Matheus Vian, Mauricio Nunes, Mírian Batista, Mlevi Lsantos, Murilo Viana, Natan Cervinski, Nathan Branco, Nicolas Teodosio, Osvaldo Neto, Otávio Carneiro, Patricia Minamizawa, Patrick Felipe, Paulo D., Paulo Tadei, Pedro Henrique, Pedro Pereira, Peterson Santos, Priscila Santos, Pydocs Pro, Pytonyc, Rafael Lopes, Rafael Romão, Rafael Veloso, Raimundo Ramos, Ramayana Menezes, Regis Santos, Renato Oliveira, Rene Bastos, Ricardo Silva, Riverfount, Rjribeiro, Robson, Robson Maciel, Rodrigo Freire, Rodrigo Oliveira, Rodrigo Quiles, Rodrigo Ribeiro, Rodrigo Vaccari, Rodrigo Vieira, Rogério Lima, Rogério Nogueira, Ronaldo Silveira, Rui Jr, Samanta Cicilia, Thaynara Pinto, Thiago Araujo, Thiago Borges, Thiago Curvelo, Tiago Minuzzi, Tony Dias, Tyrone Damasceno, Uadson Emile, Valcilon Silva, Valdir Tegon, Vcwild, Vicente Marcal, Vinicius Stein, Vladimir Lemos, Walter Reis, Willian Lopes, Wilson Duarte, Wilson Neto, Xico Silvério, Yuri Fialho, Zeca Figueiredo

Obrigado você



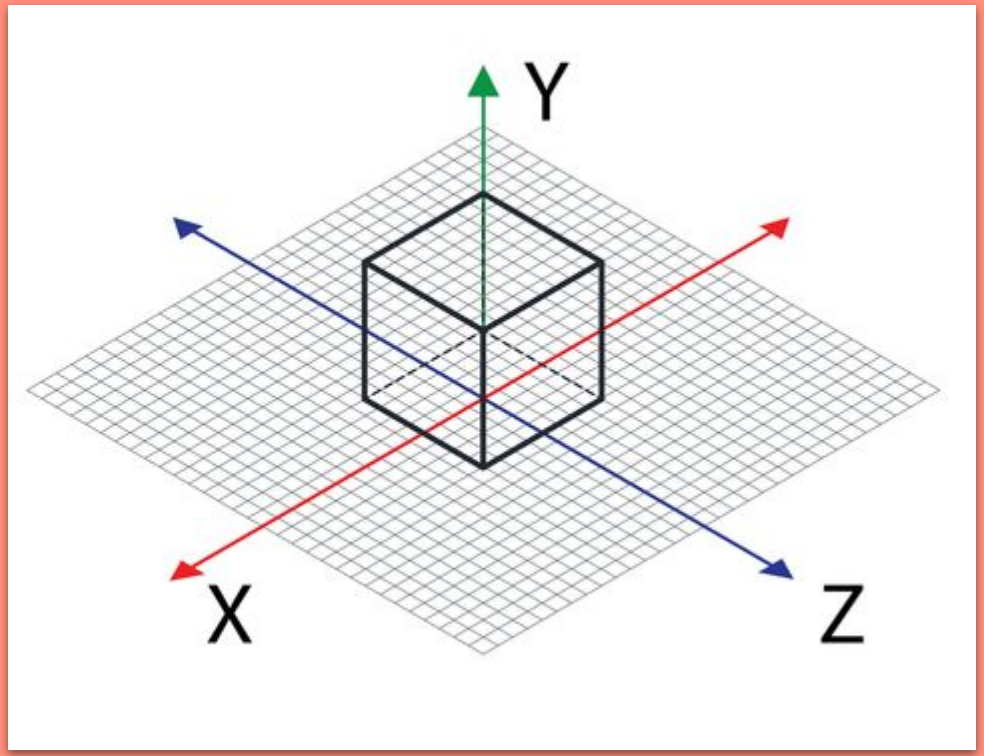
https://github.com/pokepetter/ursina/blob/master/samples/fps.py

Alguns conceitos básicos

3D

Conceitos básicos sobre 3D

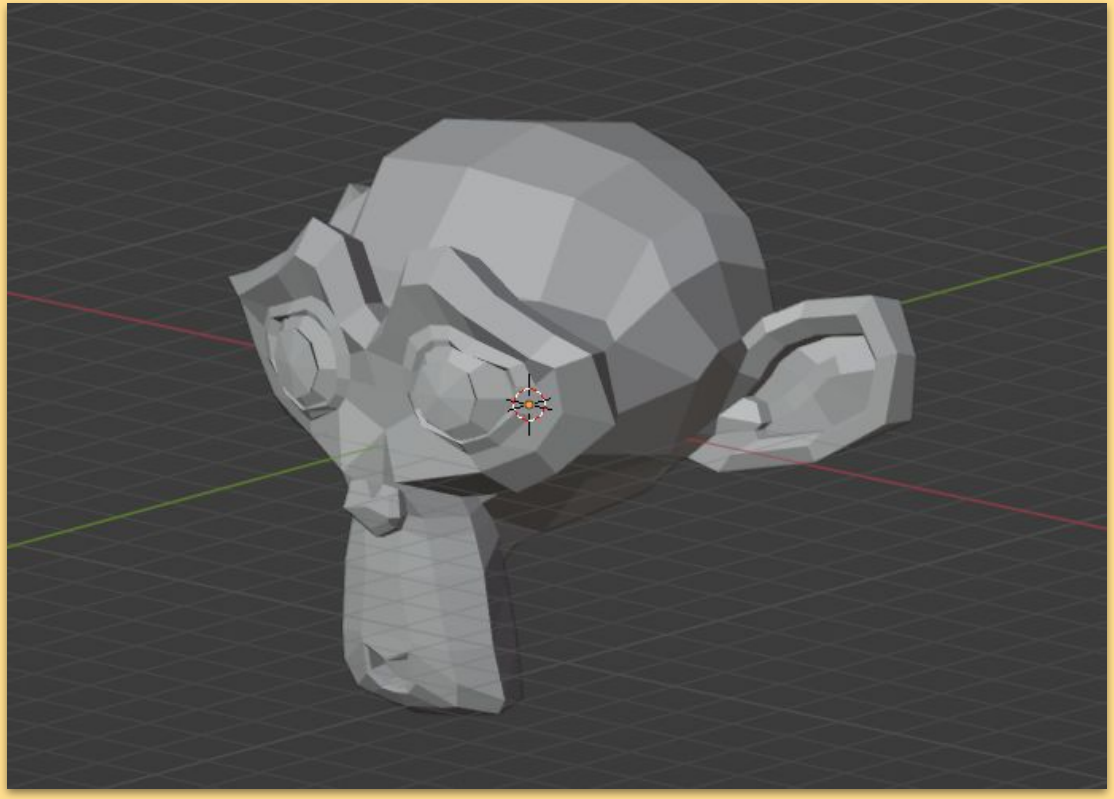
3D é um termo usado para representar 3 dimensões.



https://www.schoolofmotion.com/blog/3d-motion-design-glossary

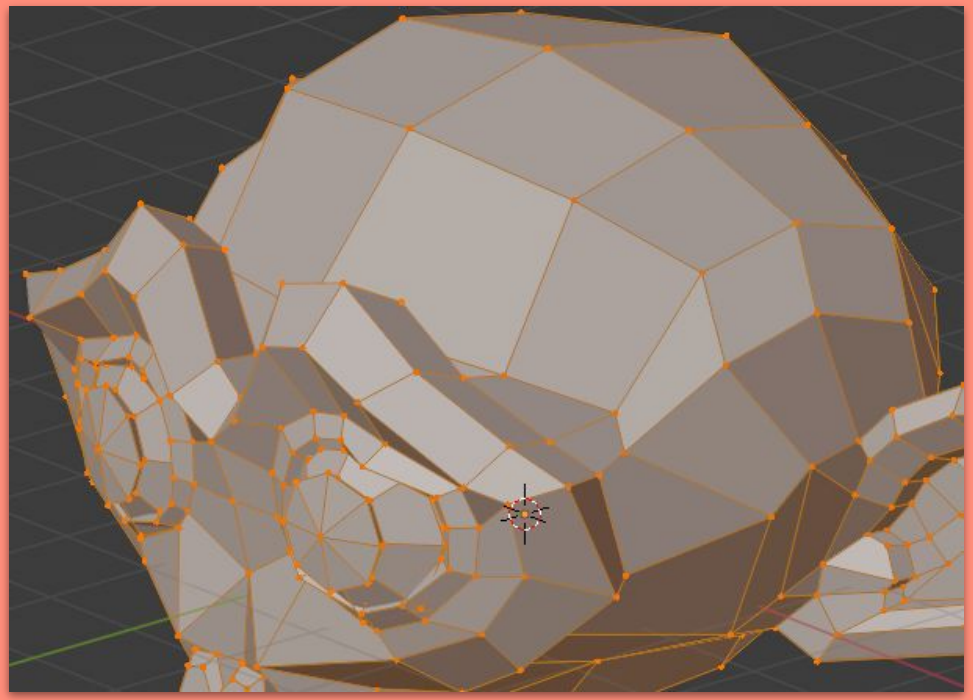
Modelos / Objetos

Qualquer objeto em 3D costumamos chamar de modelo ou objeto. Por exemplo, aqui está a Suzanne



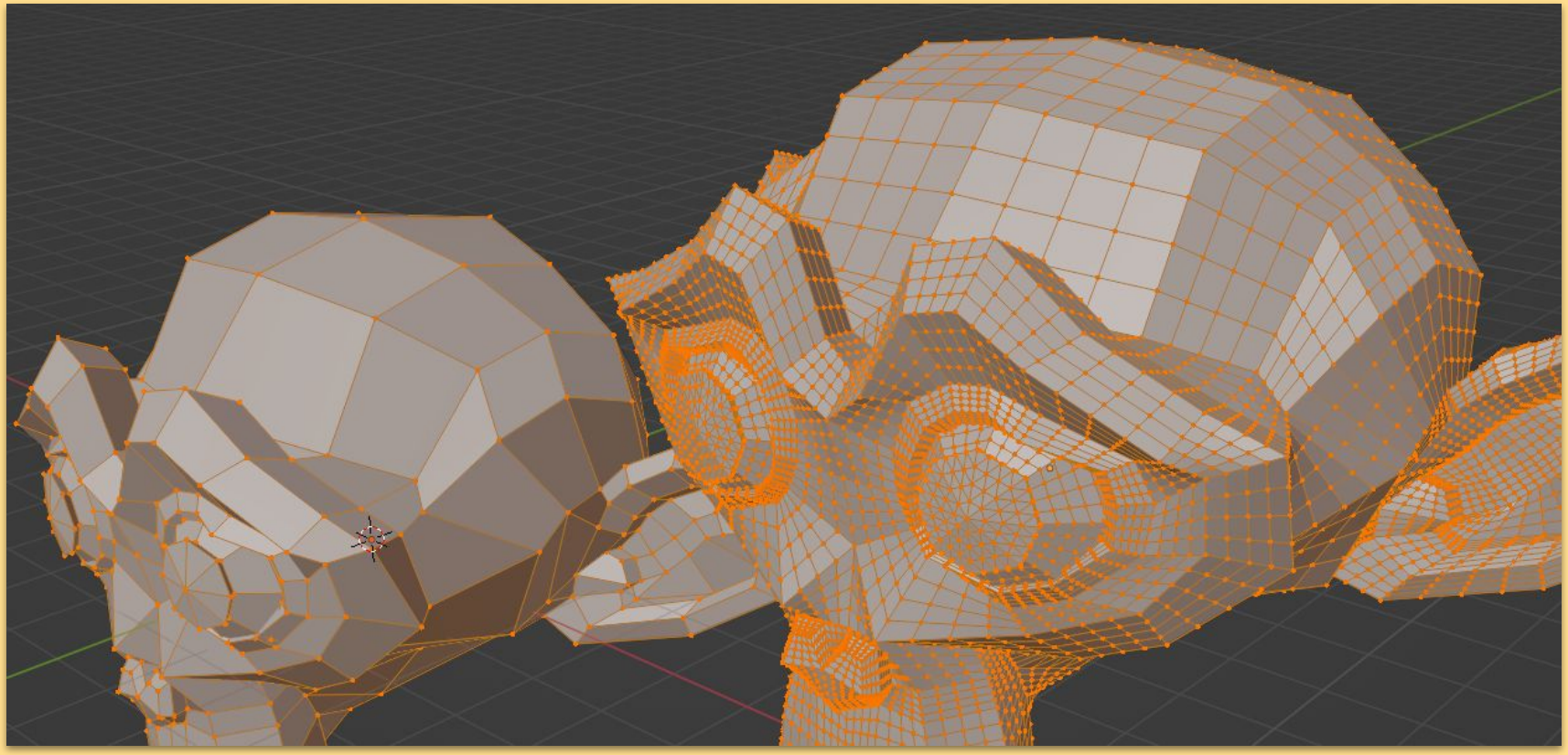
Mesh / Malha

Mesh é uma malha poligonal. Que liga vértices a arestas que criam faces poligonais.



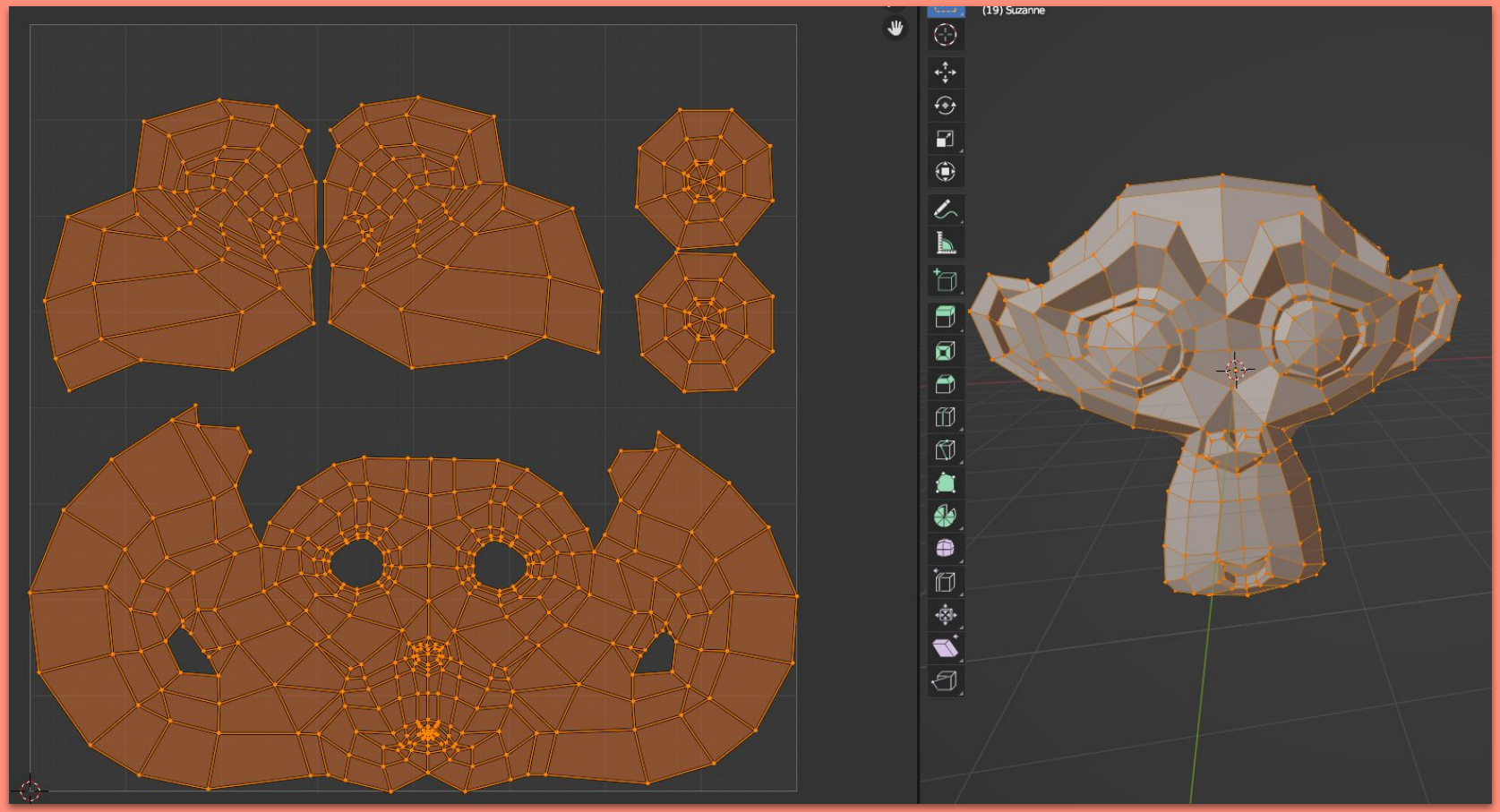
Topologia

Propriedades geométricas da malha



Mapeamento UV

Um mapa que contém uma versão achatada e desembrulhada da geometria 3D. O mapa UV permite que **texturas** 2D sejam colocadas em partes correspondentes da malha.



Textura

Uma imagem 2D (bitmap ou procedural) usada no mapeamento de um objeto 3D e descreve as várias propriedades da superfície, incluindo altura, normais, **reflexão**.

Shaders

Shader é um algorítimo que explica como o objeto/modelo sera renderizado em tempo de execução. Se sofrerá mudança com iluminação, por exemplo. Ou se for transparente. Coisas nesse sentido!

Conhecendo as entidades

Ursina

Ursina

- Criado por Petter Amland
- Biblioteca criada para desenvolvimentos de jogos com foco em 3D
- Licença: **MIT**
- Baseado no Panda3D [https://www.panda3d.org/]
  - Escrito em C++ e que também suporta python
  - **Vale uma live sobre Panda3D?**
- Release inicial: Maio de 2020
- Release atual: **5.3.0**
- Suporta modelos exportados do blender (.blend)
- Suporte multiplataforma: Gnu/Linux, MacOS e Windows

pip install ursina

Instalação

Ursina

Toda a programação no Ursina é baseada em dois conceitos:

- Entidades - **Entity**
- Atualizações - **update**
- Eventos - **input**

Basicamente podemos desenvolver qualquer coisa no ursina usando esses três conceitos.

Ursina

Toda a programação no ursina é baseada em dois conceitos:

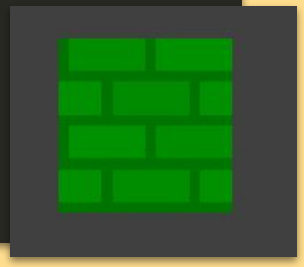
- Entidades - **Entity**
  - A entidade é basicamente a representação de qualquer objeto no ursina.
- Atualizações - **update**
  - Os updates controlam a ação das entidades em relação ao tempo
- Eventos - **input**
  - Os eventos são como as entidades reagem à interação de algum input, como mouse ou teclado

Entidades

Entidades são usadas para representar todo e qualquer objeto no Ursina. Embora possam fazer quase tudo, vamos nos forcar nos conceitos principais primeiro:

- Model: Modelo é o que define a forma do objeto
- Texture: A aparência exterior do objeto
- Color: A cor do objeto

```
1  from ursina import Entity, color
2
3  e = Entity(
4    model='cube',
5    texture='brick',
6    color=color.green
7  )
```



Objeto Ursina

Para podermos visualizar nosso cubo, precisamos criar uma classe de jogo, isso pode ser feito instanciando a classe **Ursina.**

```
1  from ursina import Entity, Ursina, color
2
3  app = Ursina()
4
5  e = Entity(
6    model='cube', texture='brick', color=color.green
7  )
8
9  app.run()
```

Objeto Ursina

Para agilizar o processo de abrir e fechar na live, vou adicionar as bordas do meu gerenciador de janelas e deixar em modo janela

uma classe de

```
1   from ursina import E      , Ursina, color
2
3   app = Ursina(borderless=False, fullscreen=False)
4
5   e = Entity(
6     model='cube', texture='brick', color=color.green
7   )
8
9   app.run()
```

O objeto Ursina

Existem outros atributos que podem ser importantes para vocês em alguns momentos, mas não vou me aprofundar nisso agora!

- **title**: Título da janela do jogo
- **fullscreen**: Bool, para ser fullscren ou não
- **size**: Tamanho da janela
- **show_ursina_splash**: Mostra a tela de abertura do Ursina
- **development_mode**: Permite fazer reload com F5
- **editor_ui_enabled**: Abre opções de desenvolvimento na tela

Pode ser que isso te ajude em algum momento xD

Voltando as Entidades

Além de propriedades básicas, precisamos entender sobre o modelo da entidade. A classe Entity, também conta com os atributos:

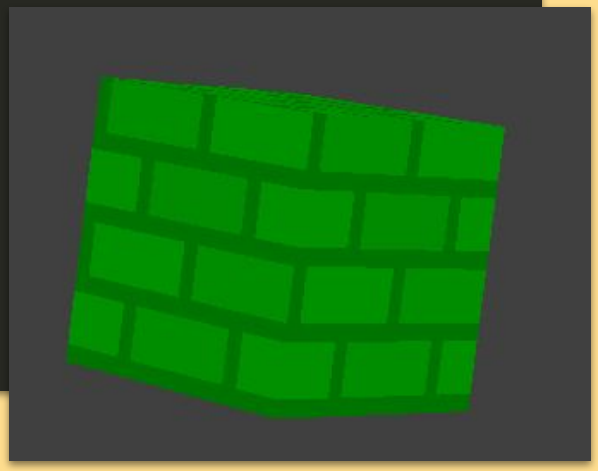
- **scale**: Define o tamanho da entidade em x, y e z
- **rotation**: Define a rotação da entidade em x, y e z
- **position**:Define a posição da entidade em x, y e z

Esses valores podem ser passados em inteiros ou em tuplas.

A tupla representa (x, y, z) e o inteiro (x * int, y * int, z * int)

Escala, posição e rotação

```python
e = Entity(
    model='cube',
    texture='brick',
    color=color.green,
    scale=2,
    rotation=(3, 35, 7),
    position=(0, -1, 0)
)
```



Movimento de câmera

Sei que essas coisas são razoavelmente difíceis de visualizar. Então, importemos a câmera do editor!

```
1   from ursina import EditorCamera
2   ...
3   EditorCamera()
4   app.run()
```

A movimentação acontece usando os botões do mouse.

Entidades

Entidades são usadas para representar todo e qualquer objeto no Ursina. Embora possam fazer quase tudo, vamos nos forcar nos conceitos principais primeiro:

- **Model**: Modelo é o que define a forma do objeto
- Texture: A aparên[illegible] do objeto
- Color: A cor d[illegible]

Existem diversos tipos de modelos prontos:

- plane
- cube
- sphere
- ...

https://www.ursinaengine.org/api_reference.html#models

Entidades

Entidades são usadas para representar todo e qualquer objeto no Ursina. Embora possam fazer quase tudo, vamos nos forcar nos conceitos principais primeiro:

- Model: Modelo é o que define a forma do objeto
- **Texture**: A aparência exterior do objeto
- Color: A c r d

Existem diversos tipos de texturas prontas:

- noise
- grass
- vignette
- ...

https://www.ursinaengine.org/api_reference.html#textures

Shaders

- **Lit With Shadows Shader**:
  - Oferece iluminação e suporte para sombras
- **Unlit Shader**:
  - Renderiza a textura diretamente sem qualquer iluminação
  - Caso não queira se preocupar com iluminação
- **Normal Shader**: Usa o mapeamento UV para dar a ilusão de detalhes
  - Detalhes 3D em objetos 2D
- **Color Shader**: Aplica uma cor a entidade
- **Texture Shader**: Aplica uma textura a entidade
- **Fog Shader**: Efeito de névoa ao ambiente
- **Sky Shader**: Renderiza o céu em um ambiente 3D.
- **Terrain Shader**: Renderiza terrenos complexos

Iluminação

Para iluminar as entidades e dar uma “imersão" no jogo. Podemos adicionar pontos de luz. Para gerar sombras e também interagir com os shaders. O Ursina provém diversos tipos de iluminação:

- **DirectionalLight**: Luz direcional é como um sol, não importa onde você esteja, a luz e as sombras terão a mesma direção
  - Única luz capaz de gerar sombras até o momento
- **PointLight**:A luz pontual é semelhante a uma lâmpada, ilumina em todas as direções a partir de um único ponto
- **AmbientLight**: O sol, ilumina todas a entidades de um mesmo ponto
- **SpotLight**: Luz spot é semelhante a uma lanterna, ilumina a partir de um único ponto em uma direção específica

Mostrar os exemplos em código. Para quem acessar depois, está na pasta luzes!

Áudio

Para usar áudio no ursina, podemos somente importar a classe de Audio.

```
1   ...
2   from ursina.audio import Audio
3   ...
4   shot = Audio('explosion.wav', autoplay=False)
5
6   shot.play()
7   ...
```

Prefabs

Existem diversos elementos pré-fabricados no Ursina, que podem ser bastante divertidos de usar:

- Sky
- EditorCamera: Que já usamos
- FirstPersonController
- ...

```python
from ursina.prefabs.editor_camera import EditorCamera
from ursina.prefabs.sky import Sky

Sky()  # o céu aparecerá
EditorCamera()  # Permite controlar a cena com o mouse
```

Dinâmico

Interações e movimentação dos objetos

Interações

No Ursina, existem dois métodos para deixar as coisas dinâmicas:

- **update**: Uma função que toda em todos os quadros (60fps)
- **input**: Que lida com a interação do teclado e mouse
- **Ações de mouse**: Interação de mouse com os objetos

Update [exemplo_update.py]

O jogo pode ter uma opção global de update ou as entidades podem ter funções específicas de update. Começamos pelo global

```
1   from ursina import time
2   from ursina.input_handler import held_keys
3
4   def update():
5       entidade.rotation_z += time.dt * 20
6       entidade.x += held_keys['right arrow'] * time.dt * 5
7       entidade.x -= held_keys['left arrow'] * time.dt * 5
```

Update em entidades [exemplo_update.py]

```python
from ursina import Entity, time
from ursina.input_handler import held_keys
from ursina.shaders import lit_with_shadows_shader

class Cubo(Entity):
    def __init__(self):
        super().__init__()
        self.model = 'cube'
        self.texture = 'brick'
        self.shader = lit_with_shadows_shader

    def update(self):
        self.rotation_z += time.dt * 20
        self.x += held_keys['right arrow'] * time.dt * 5
        self.x -= held_keys['left arrow'] * time.dt * 5
```

Inputs [audio.py]

Inputs diferem de updates. Nesse caso, eles esperam pela ação de alguém. É ideal para manipulação dos objetos ou exibir fontes sonoras. Pois acontecem somente quando pressionarmos um botão.

```
1  from ursina import Audio, Ursina
2  app = Ursina(broderless=False)
3  audio = Audio('explosion.wav')
4
5  def input(key):
6      if key == 'space':
7          audio.play()
8
9  app.run()
```

Interações de mouse [mouse.py]

As interações de mouse necessitam de uma entidade **(com colisão)** e uma função envolvida. Existem algumas ações possíveis:

- on_click
- on_double_click
- on_mouse_enter
- on_mouse_exit

```python
from ursina import Ursina, Entity, time
app = Ursina(borderless=False)

def rotate():
    e.rotation_z += time.dt * 45

e = Entity(model='cube', collider='box', on_click=rotate)

app.run()
```

Colisão

Ações de colisão são quando dois objetos colidem entre si. Ou seja, o mouse em algo, um objeto em outro objeto. O Ursina tem diversos tipos de colisores. Um para cada tipo de malha.

- BoxCollider
- SphereCollider
- MeshCollider: Para usar a forma da malha todo como colisão

```
1  from ursina import Ursina, Entity, time
2  app = Ursina(borderless=False)
3
4  def rotate():
5      e.rotation_z += time.dt * 45
6
7  e = Entity(model='cube', collider='box', on_click=rotate)
8
9  app.run()
```

Algumas ferramentas úteis

Algumas coisas que não cobrimos, mas que podem ser completamente úteis:

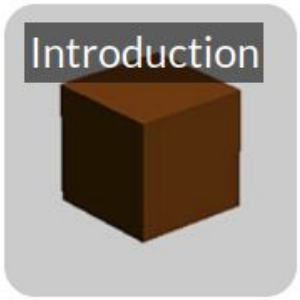
- **ursinastuff**: Utilitários importantes e prontos que podem te ajudar
  - **destroy**
  - **invoke**
- **ursinamath**: Alguns cálculos prontos ou formas simples de fazer
  - **distance**, distance_2d, distance_xz
  - **clamp**: Para limitar o espaço de um objeto
- **colision**: Módulo de colisões
  - **raycast**: Detecta uma colisão e retorna os objetos e suas posições
- **animation**: Animações prontas
  - **Animator**: Cria animações bem legais
  - **curve**: Curva para animações

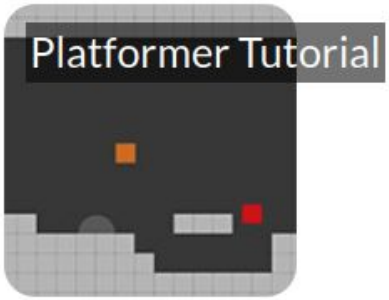
Nãso cobrimos, pois não deu tempo

Existem alguns módulos incríveis no Ursina que não cobriremos por falta de tempo:

- **UI**: Componentes para gerar interfaces
  - Botões
  - Sliders
  - FileBrowser
  - ....
- **Procedural Models**: Fábrica para modelos geométricos
  - Cones, Pipes, Terrenos e etc..
- **Camera**: É possível manipular e trabalhar a camera
- **Scene**: É possível maniuplar mais de uma cena
- **Text**:É possível trabalhar com entidades de texto 2D e 3D

Tutorials





Example Projects

https://www.ursinaengine.org/documentation.html

Vamos codar um joguinho até dar o tempo da live e se der tempo.

Podemos fazer uma live só codando um jogo? Quem é a favor, se manifeste no chat ou nos comentários!

Bora jogar até cansar!?

picpay.me/dunossauro

apoia.se/livedepython

pix.dunossauro@gmail.com

Ajude o projeto <3

Referências

- **Glossário 3d**: schoolofmotion.com/blog/3d-motion-design-glossary
- **Documentação do Ursina**: ursinaengine.org/documentation.html
- **API do Ursina**: ursinaengine.org/api_reference.html
- **Documentação do blender**: docs.blender.org