Minicurso de Introdução à Programação de Jogos 2D

(porque títulos gigantes são legais*)

Paulo Bruno de Sousa Serafim
Arthur Carvalho Walraven

*não

Sumário

Aspectos visuais

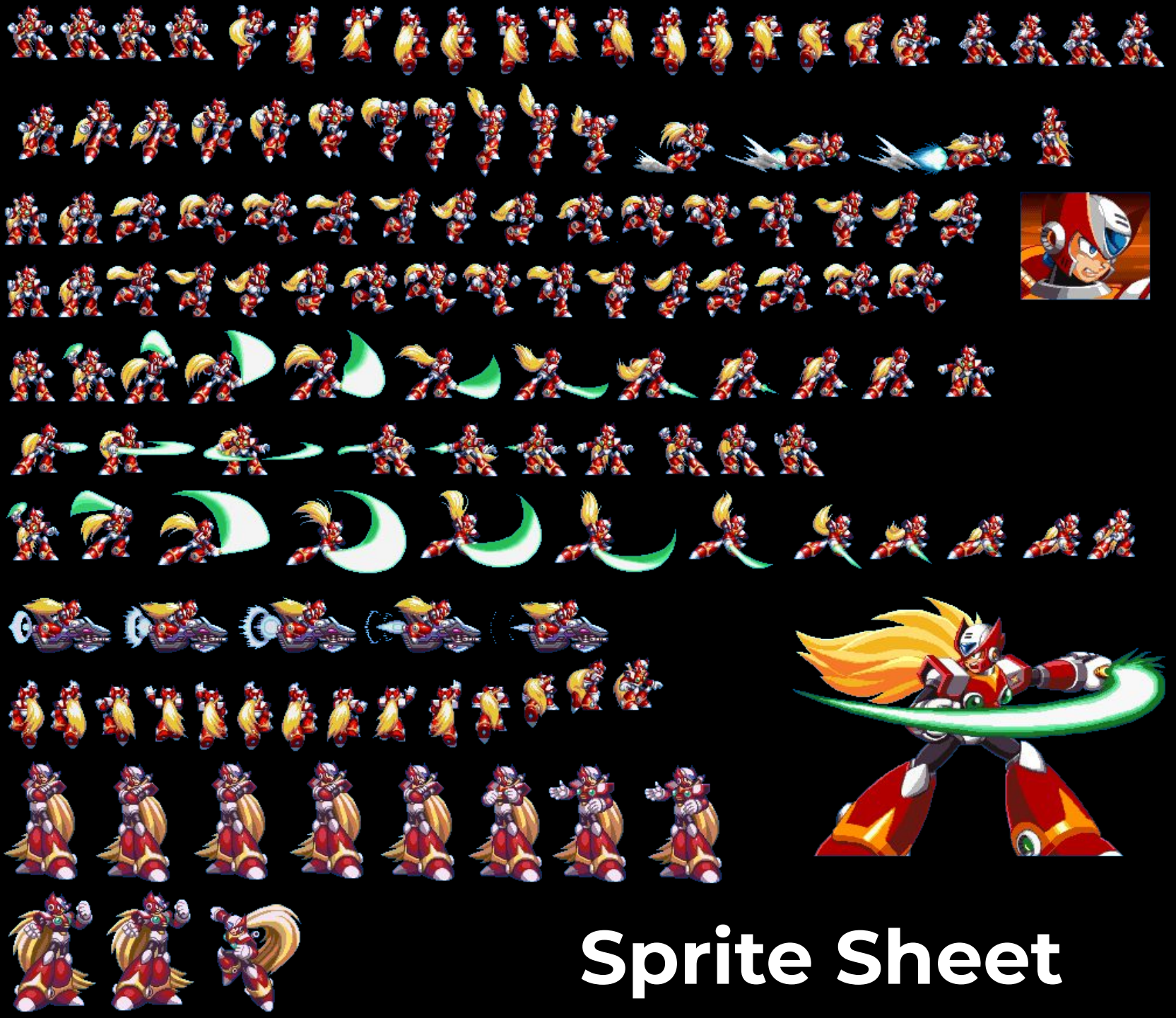

LICENCIADO PELA
Nintendo
SNS-AFZE-USA
MADE IN JAPAN
DISNEY'S
Magical Quest 3
STARRING Mickey & Donald
CAPCOM
SUPER NINTENDO
V.E.N. GAMES
1 30 60 90 120 180 13
2 0 1 2 3 4 5 6 14
3 4 5 6 7 8 8 9 15
4 5 6 7 8 9 10 11 12

Isso é um *sprite*

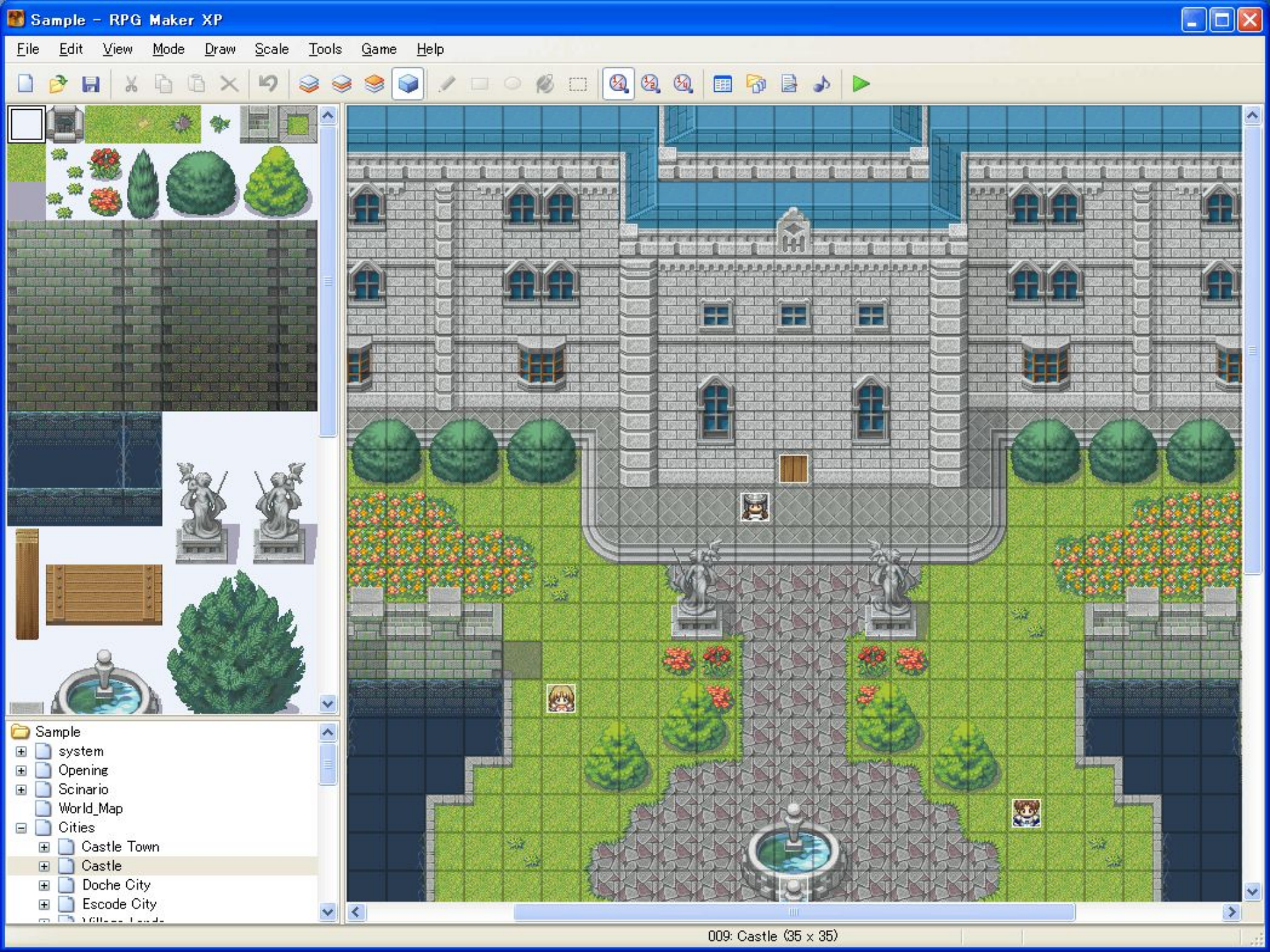
[0] [1] [2] [3] [4] [5] [6] [7]

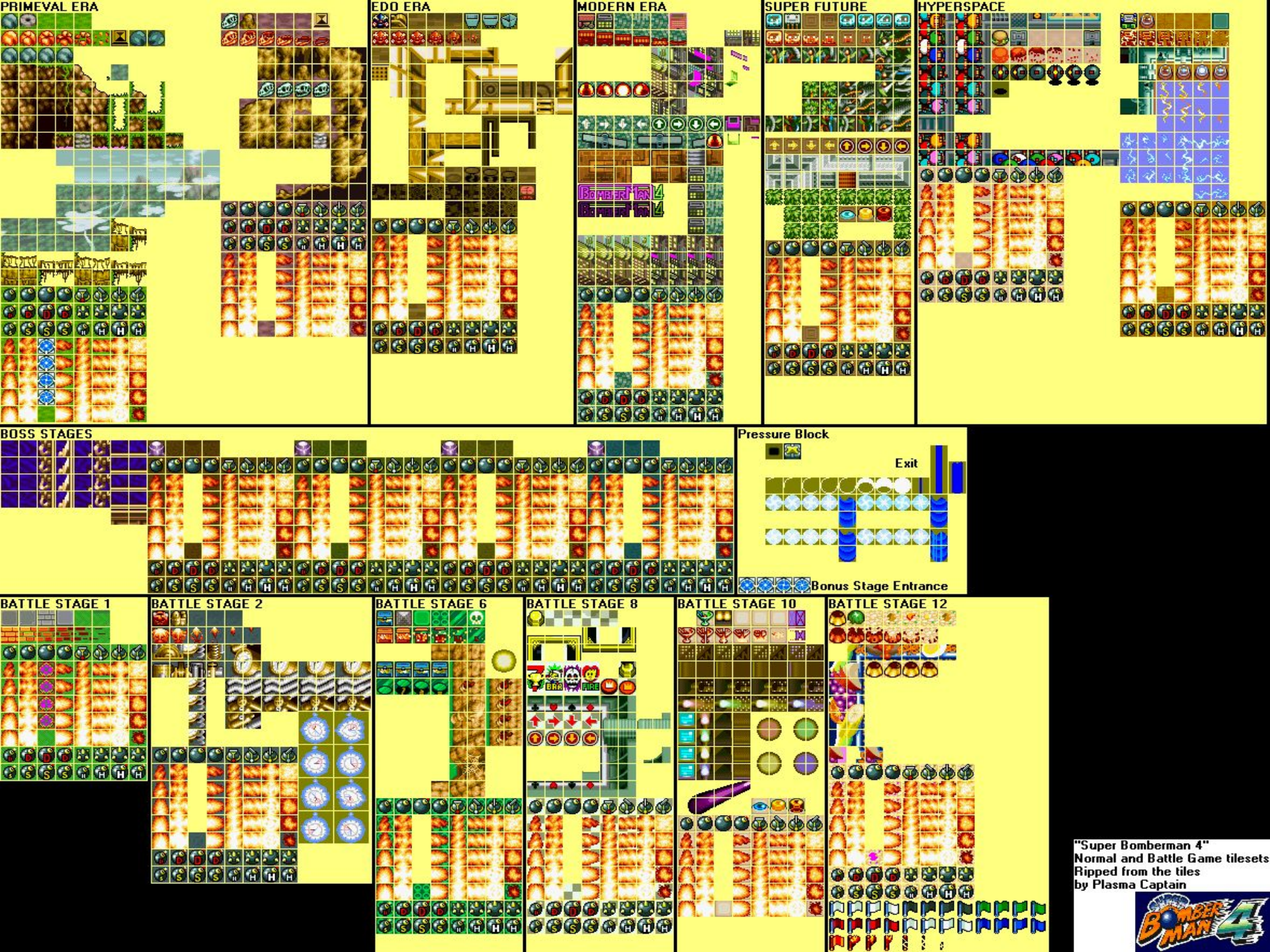
[0] [1] [2] [3] [4] [5] [6] [7]

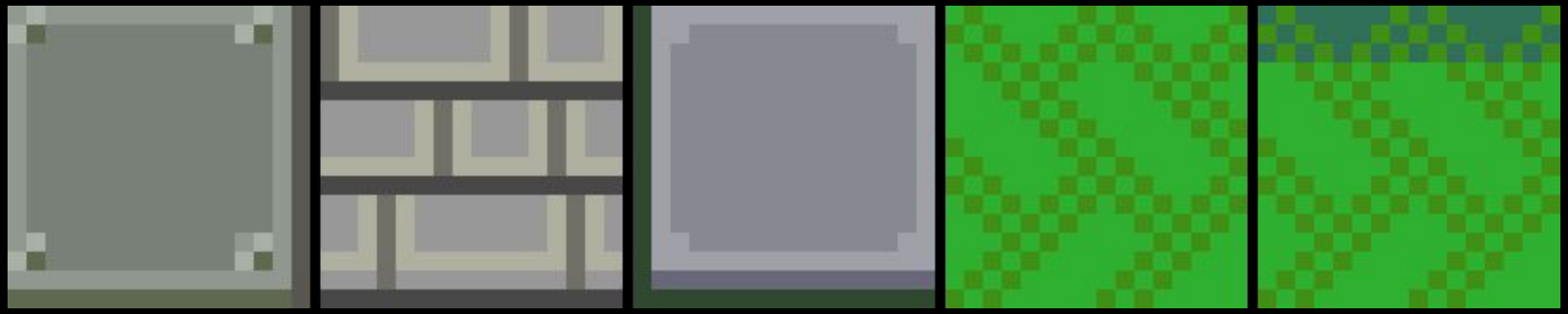
```
frame = game_timer % sprites_number
```

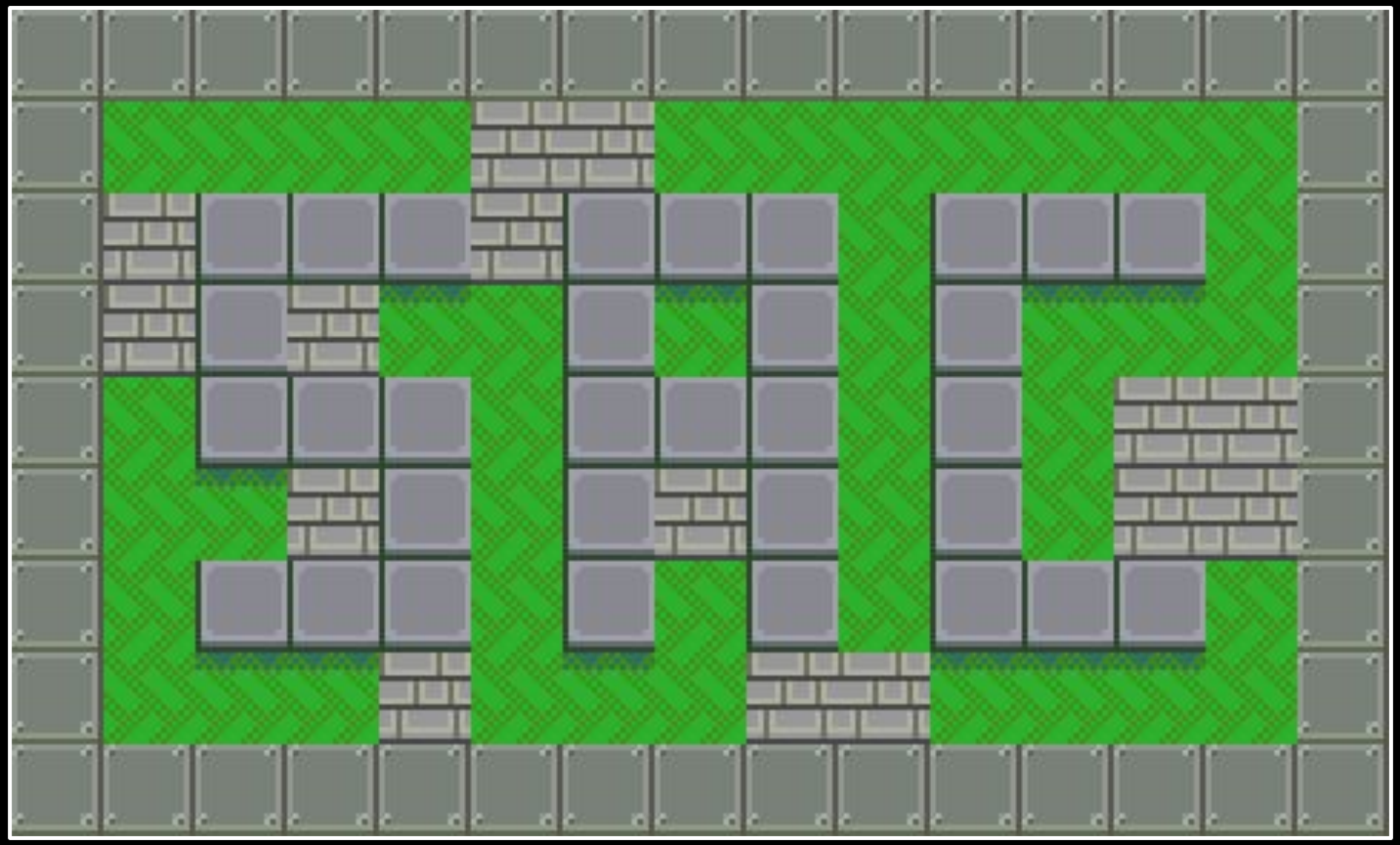

Sprite Sheet

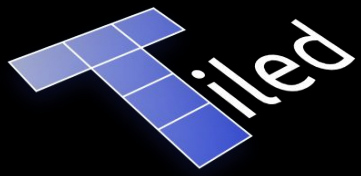

Tiling

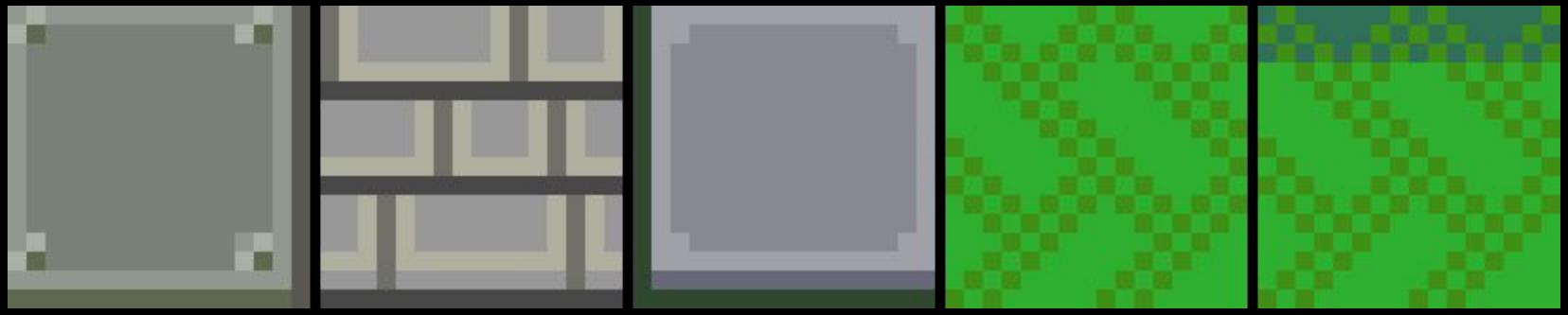

Sample - RPG Maker XP
File Edit View Mode Draw Scale Tools Game Help
Sample
system
Opening
Scinario
World_Map
Cities
Castle Town
Castle
Doche City
Escode City
009: Castle (35 x 35)


PRIMEVAL ERA
EDO ERA
MODERN ERA
SUPER FUTURE
HYPERSPACE
BOSS STAGES
Pressure Block
Exit
Bonus Stage Entrance
BATTLE STAGE 1
BATTLE STAGE 2
BATTLE STAGE 6
BATTLE STAGE 8
BATTLE STAGE 10
BATTLE STAGE 12
"Super Bomberman 4"
Normal and Battle Game tilesets
Ripped from the tiles
by Plasma Captain



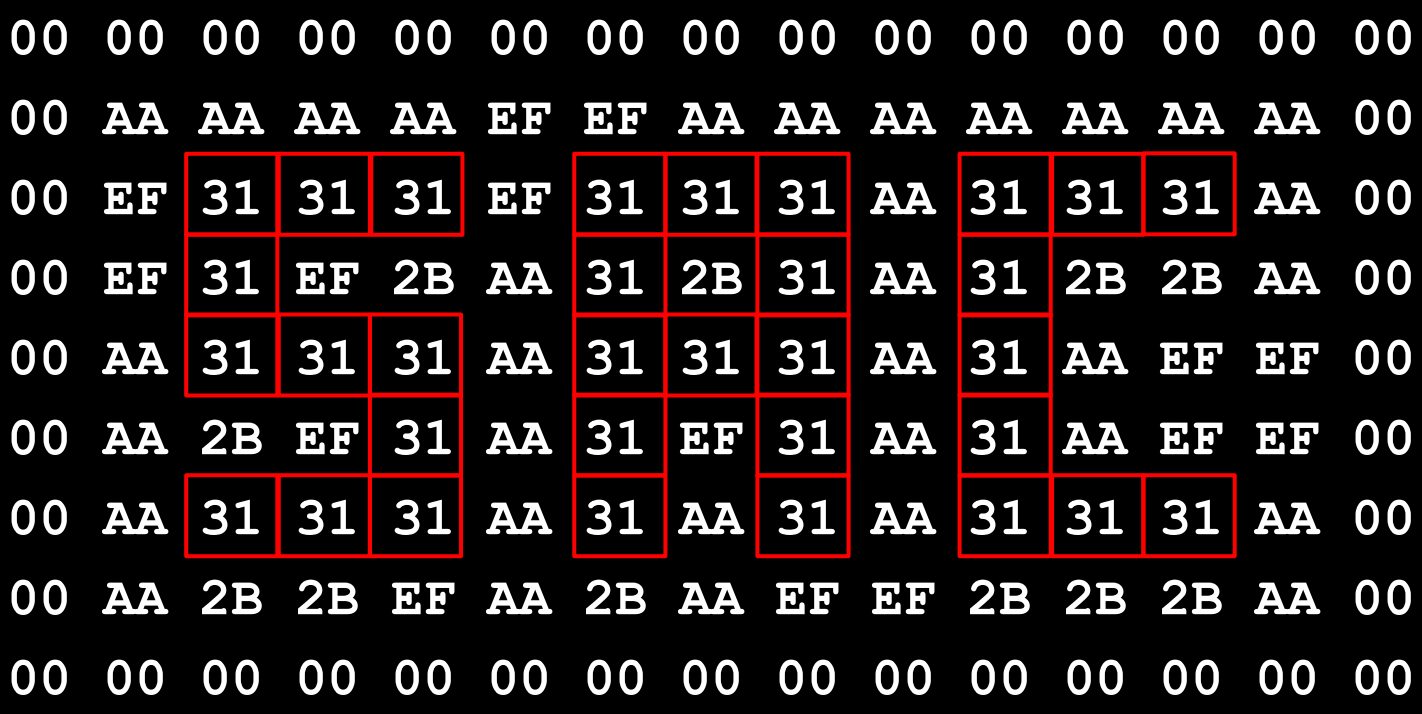


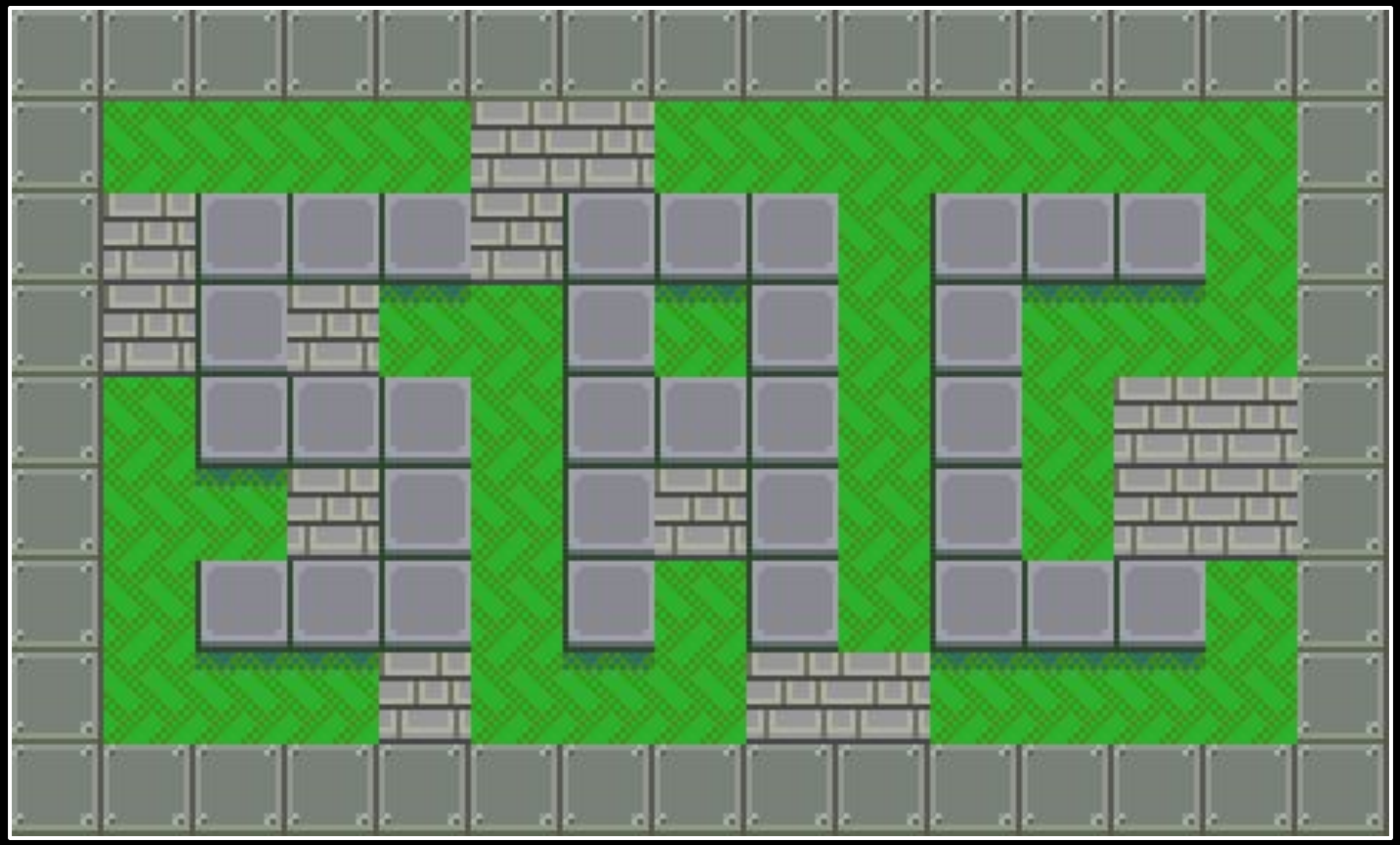

Tiled

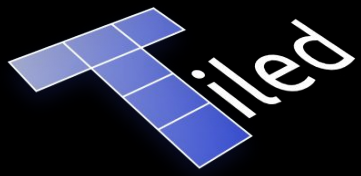


00 EF 31 AA 2B

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 AA AA AA AA EF EF AA AA AA AA AA AA AA 00
00 EF 31 31 31 EF 31 31 31 AA 31 31 31 AA 00
00 EF 31 EF 2B AA 31 2B 31 AA 31 2B 2B AA 00
00 AA 31 31 31 AA 31 31 31 AA 31 AA EF EF 00
00 AA 2B EF 31 AA 31 EF 31 AA 31 AA EF EF 00
00 AA 31 31 31 AA 31 AA 31 AA 31 31 31 AA 00
00 AA 2B 2B EF AA 2B AA EF EF 2B 2B 2B AA 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```


00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 AA AA AA AA EF EF AA AA AA AA AA AA AA 00
00 EF 31 31 31 EF 31 31 31 AA 31 31 31 AA 00
00 EF 31 EF 2B AA 31 2B 31 AA 31 2B 2B AA 00
00 AA 31 31 31 AA 31 31 31 AA 31 AA EF EF 00
00 AA 2B EF 31 AA 31 EF 31 AA 31 AA EF EF 00
00 AA 31 31 31 AA 31 AA 31 AA 31 31 31 AA 00
00 AA 2B 2B EF AA 2B AA EF EF 2B 2B 2B AA 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00



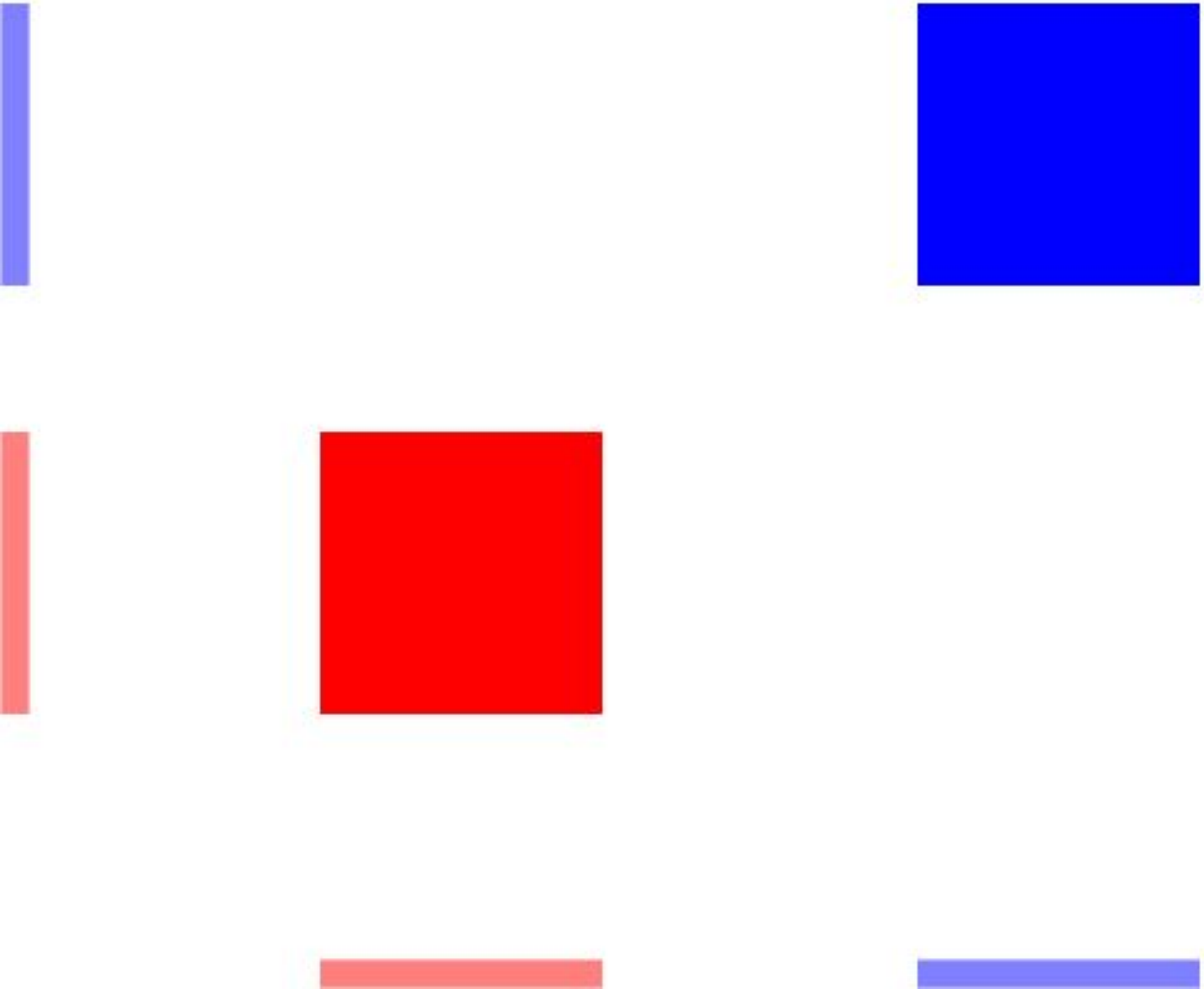

Tiled

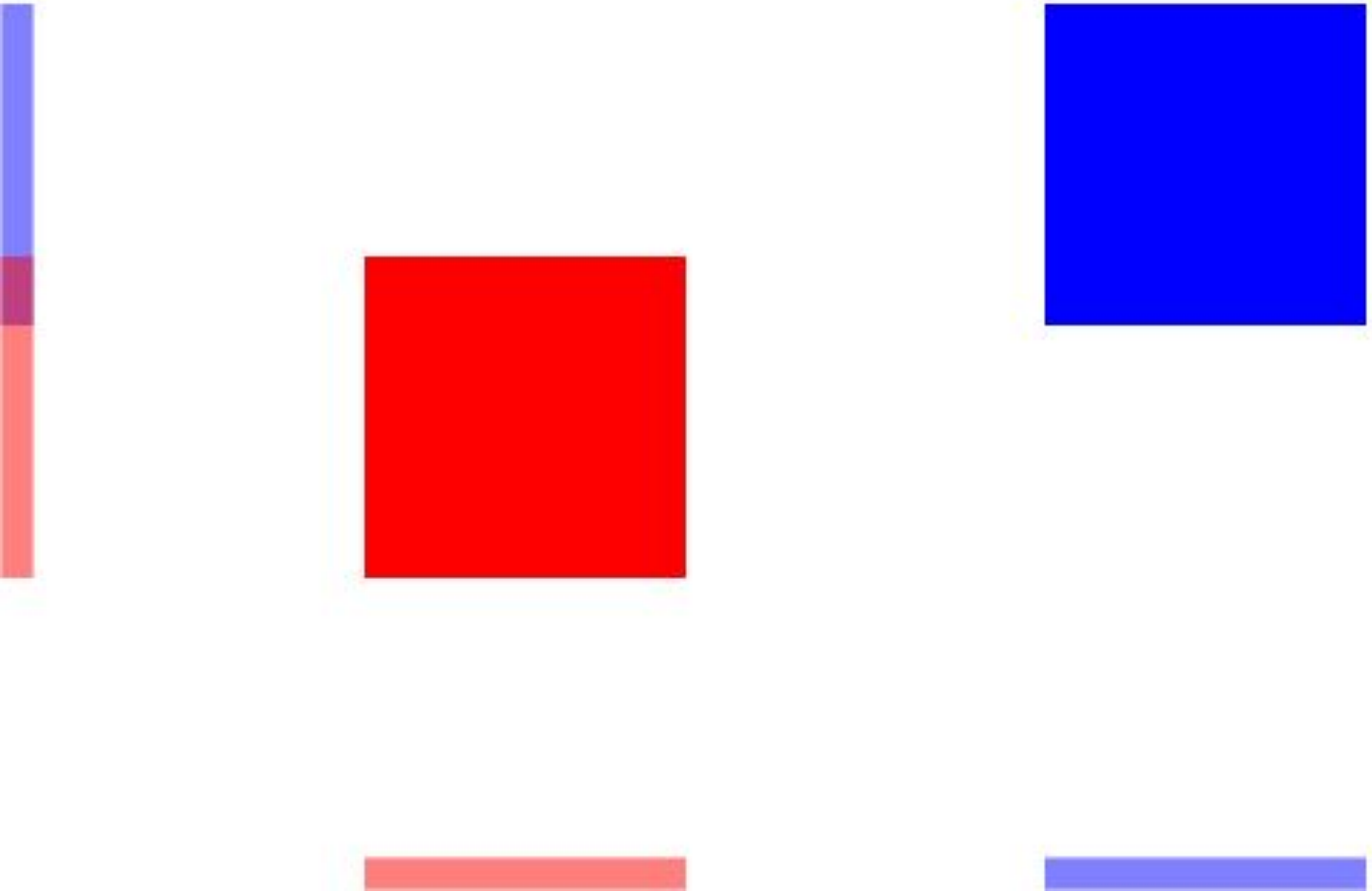
```
level[15][9] = {
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    00 AA AA AA AA EF EF AA AA AA AA AA AA AA 00
    00 EF 31 31 31 EF 31 31 31 AA 31 31 31 AA 00
    00 EF 31 EF 2B AA 31 2B 31 AA 31 2B 2B AA 00
    00 AA 31 31 31 AA 31 31 31 AA 31 AA EF EF 00
    00 AA 2B EF 31 AA 31 EF 31 AA 31 AA EF EF 00
    00 AA 31 31 31 AA 31 AA 31 AA 31 31 31 AA 00
    00 AA 2B 2B EF AA 2B AA EF EF 2B 2B 2B AA 00
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
}
```

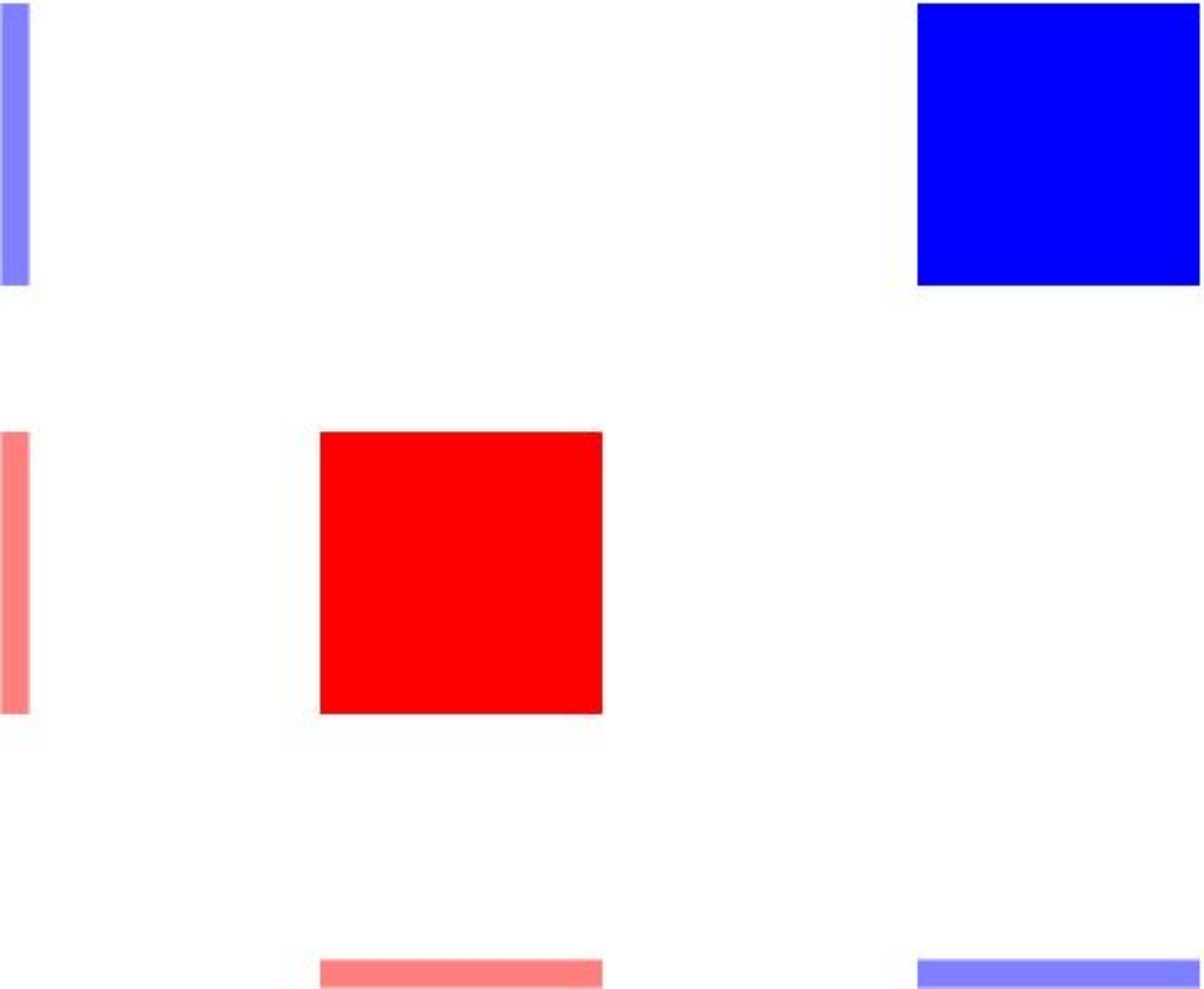

Colisão 2D

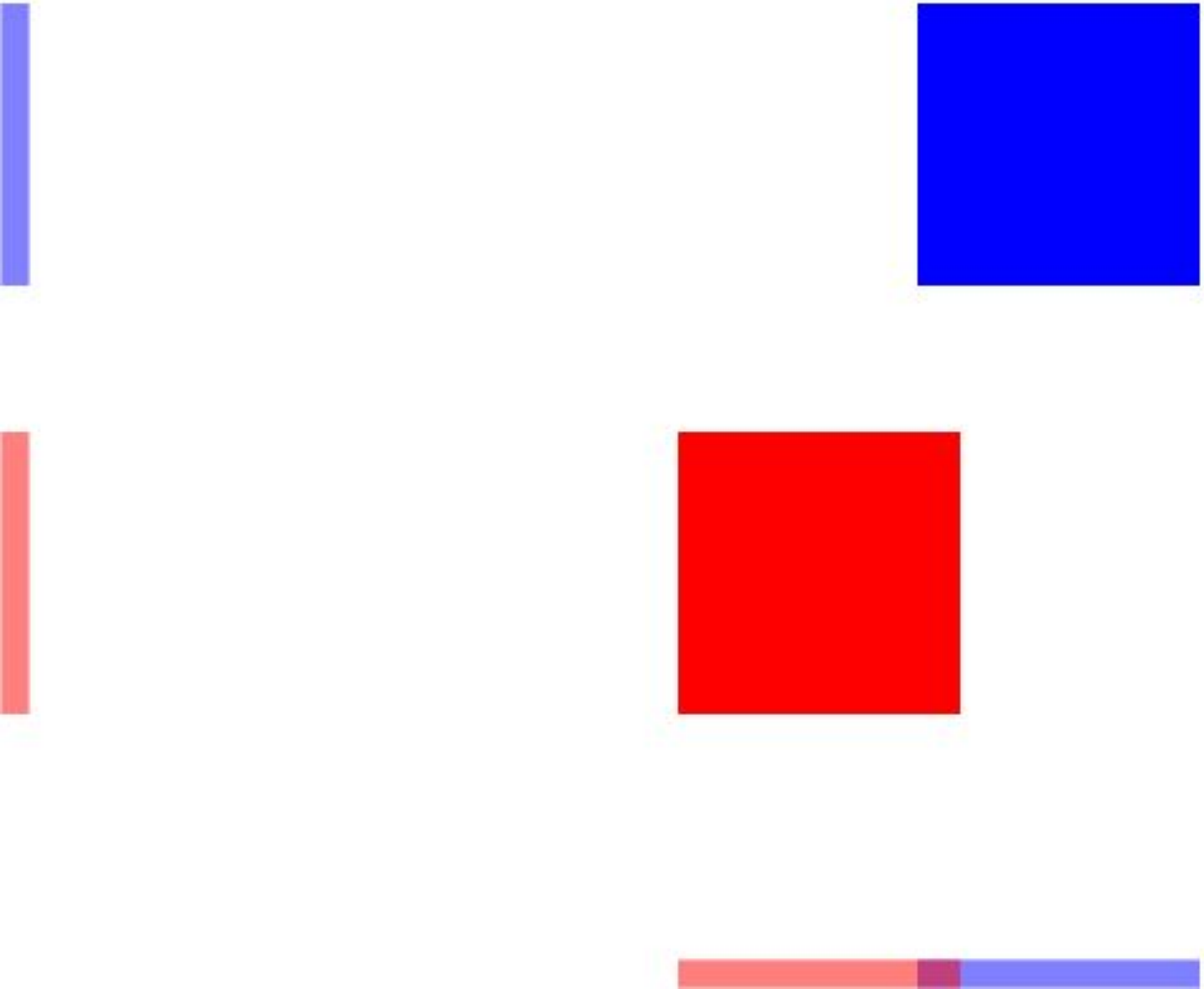

1UP CORMANO

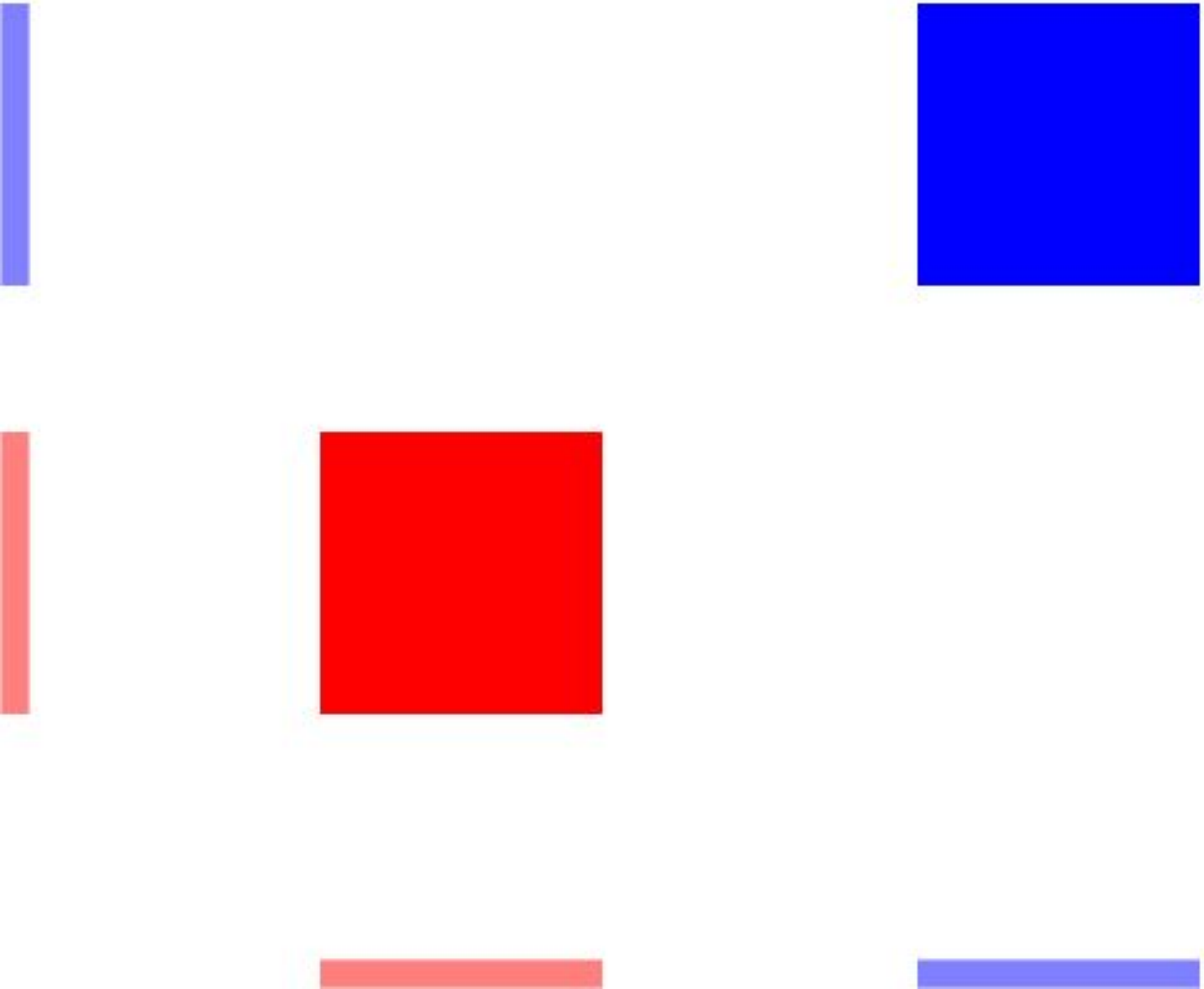

1UP CORMANO

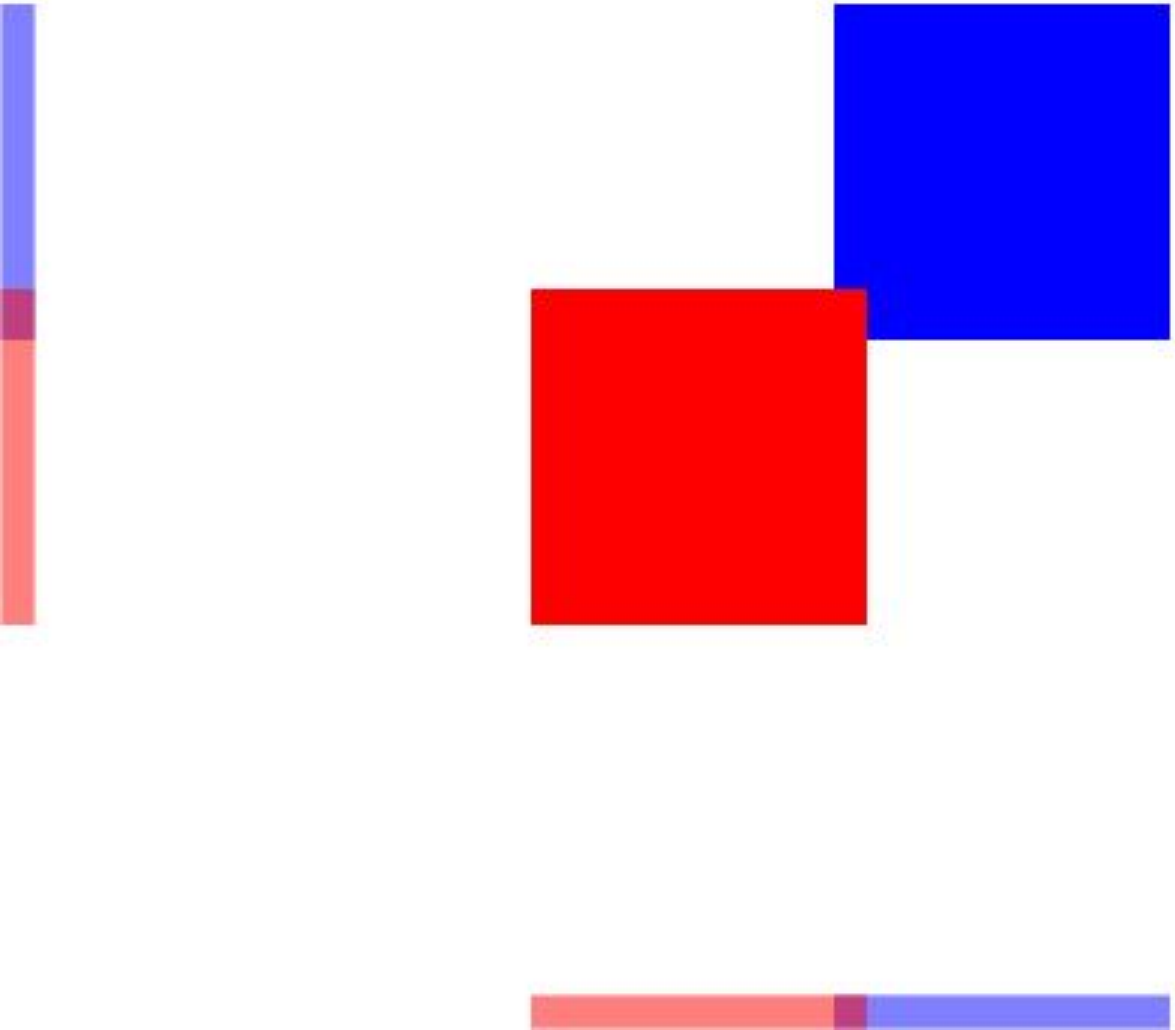


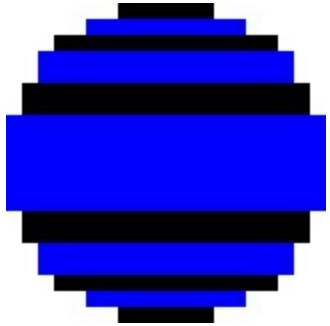


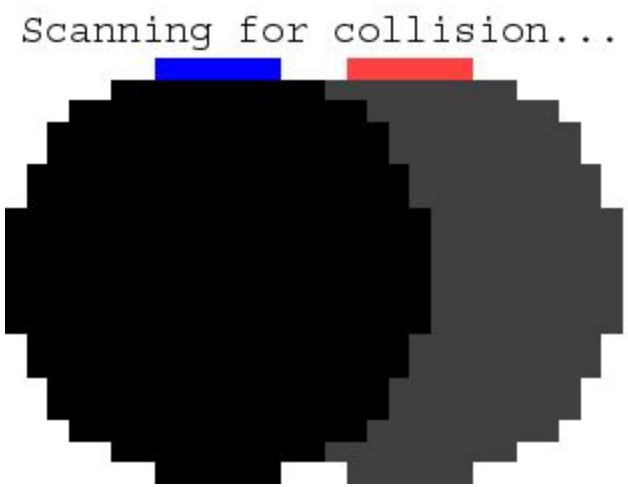










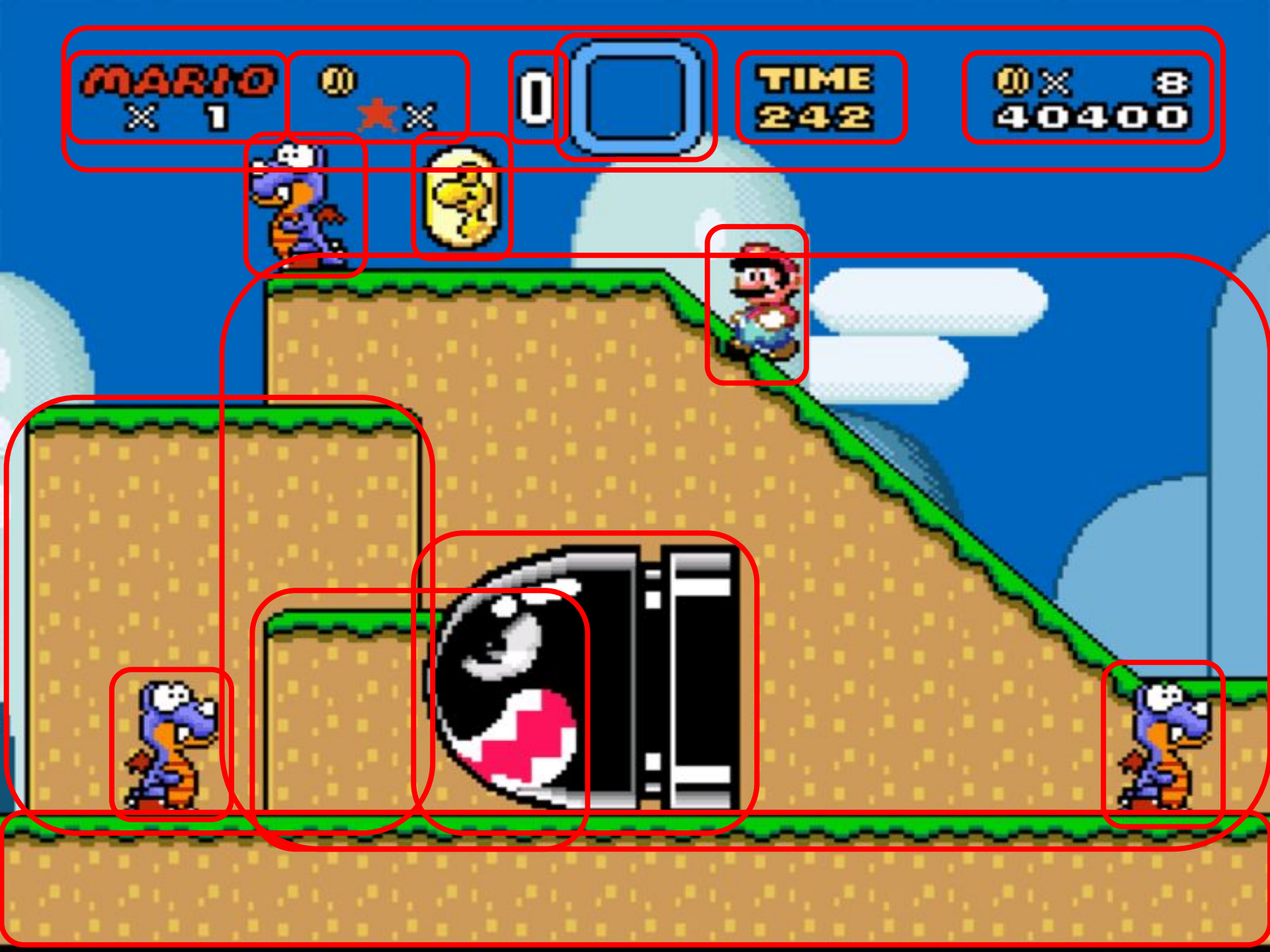

Scanning for collision...

Algumas opções de máscara de colisão:

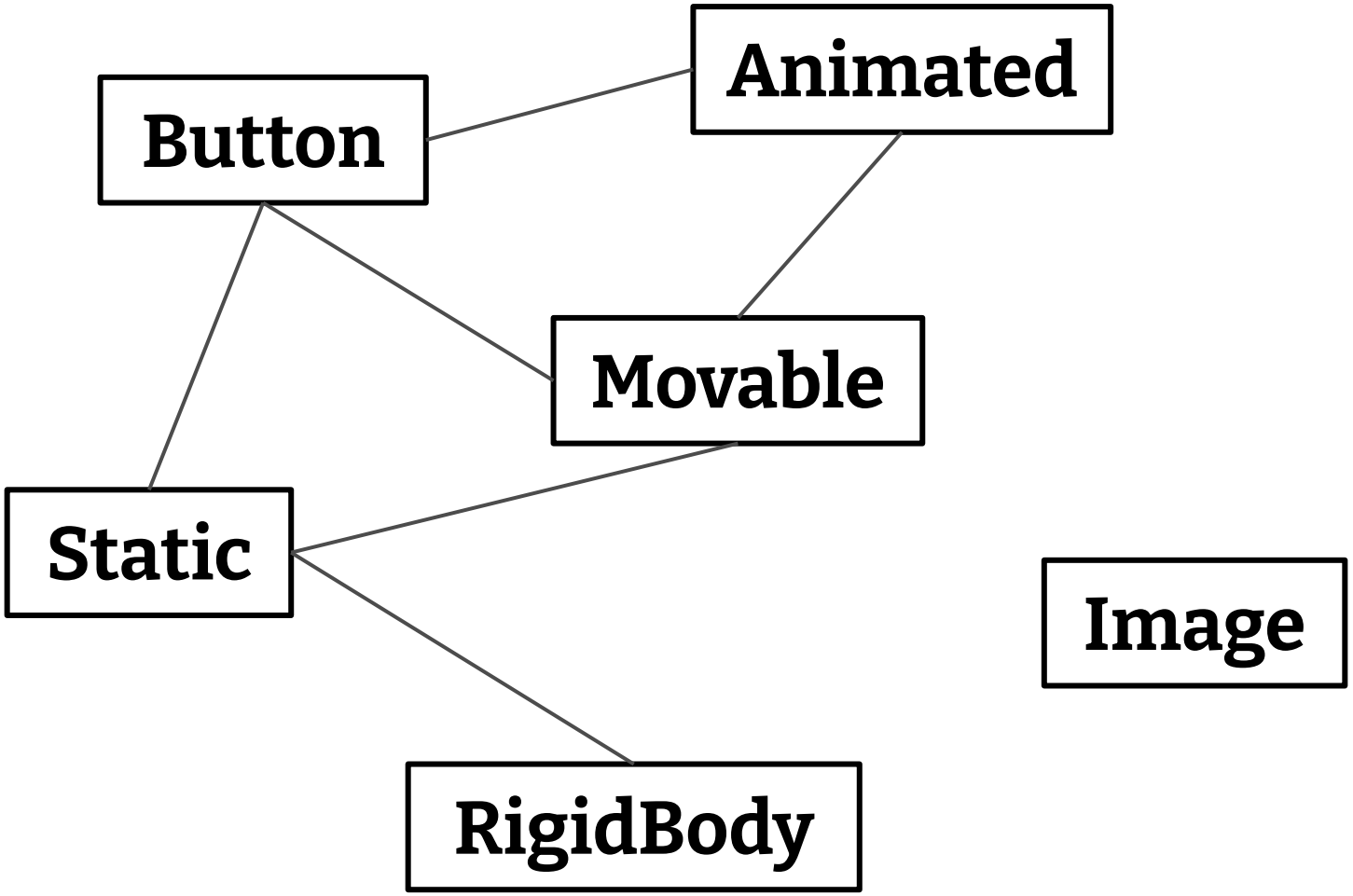
Game Objects

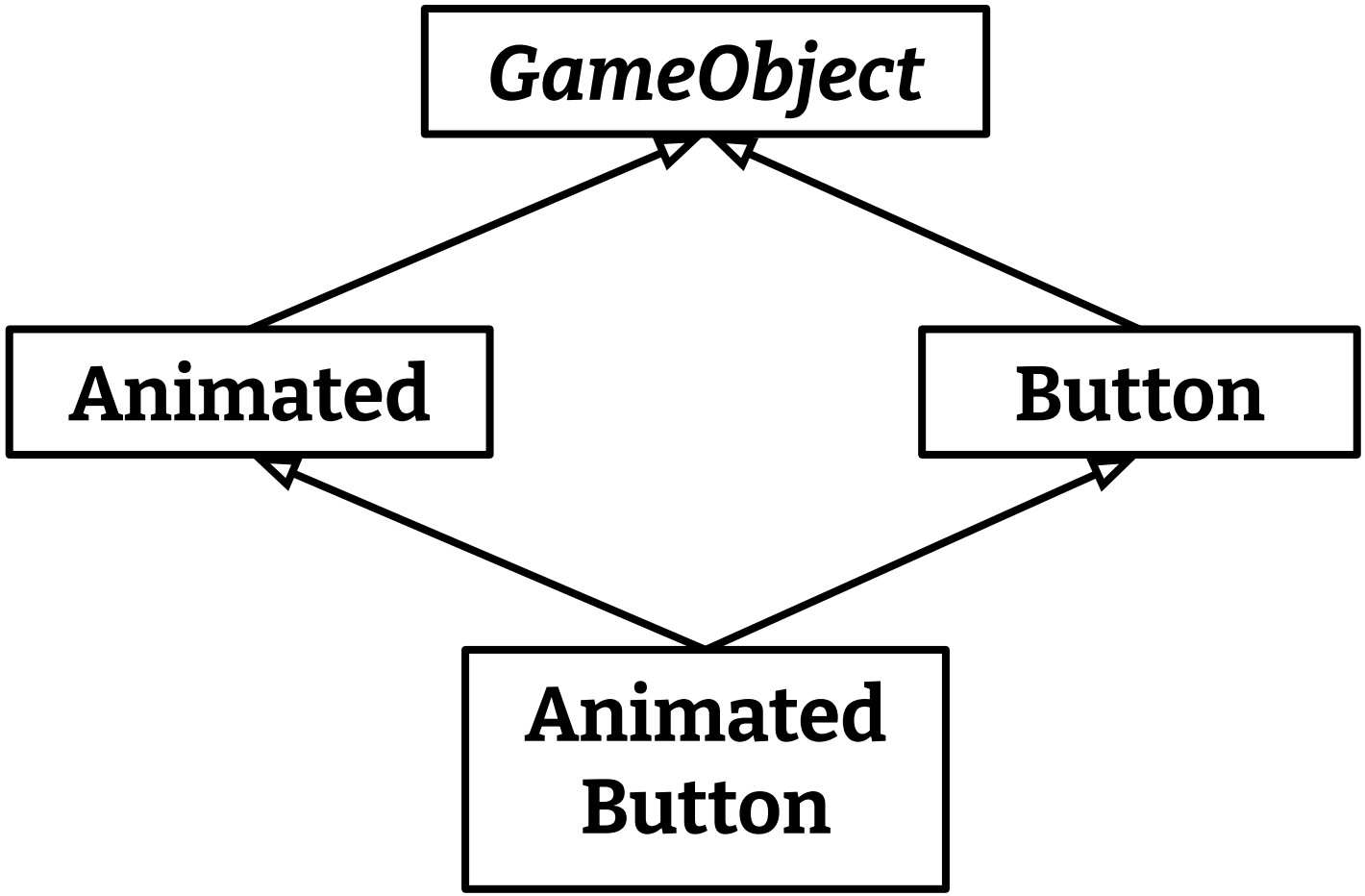
“*Qualquer coisa* que esteja no universo do jogo”


MARIO
× 1
× 0
TIME
242
× 8
40400


MARIO
× 1
× 0
TIME
242
× 8
40400


Clicável
Animação
Áudio
Movimento
Imagem
Física
Colisor


Animated
Button
Movable
Static
Image
RigidBody


GameObject
Animated
Button
Animated
Button

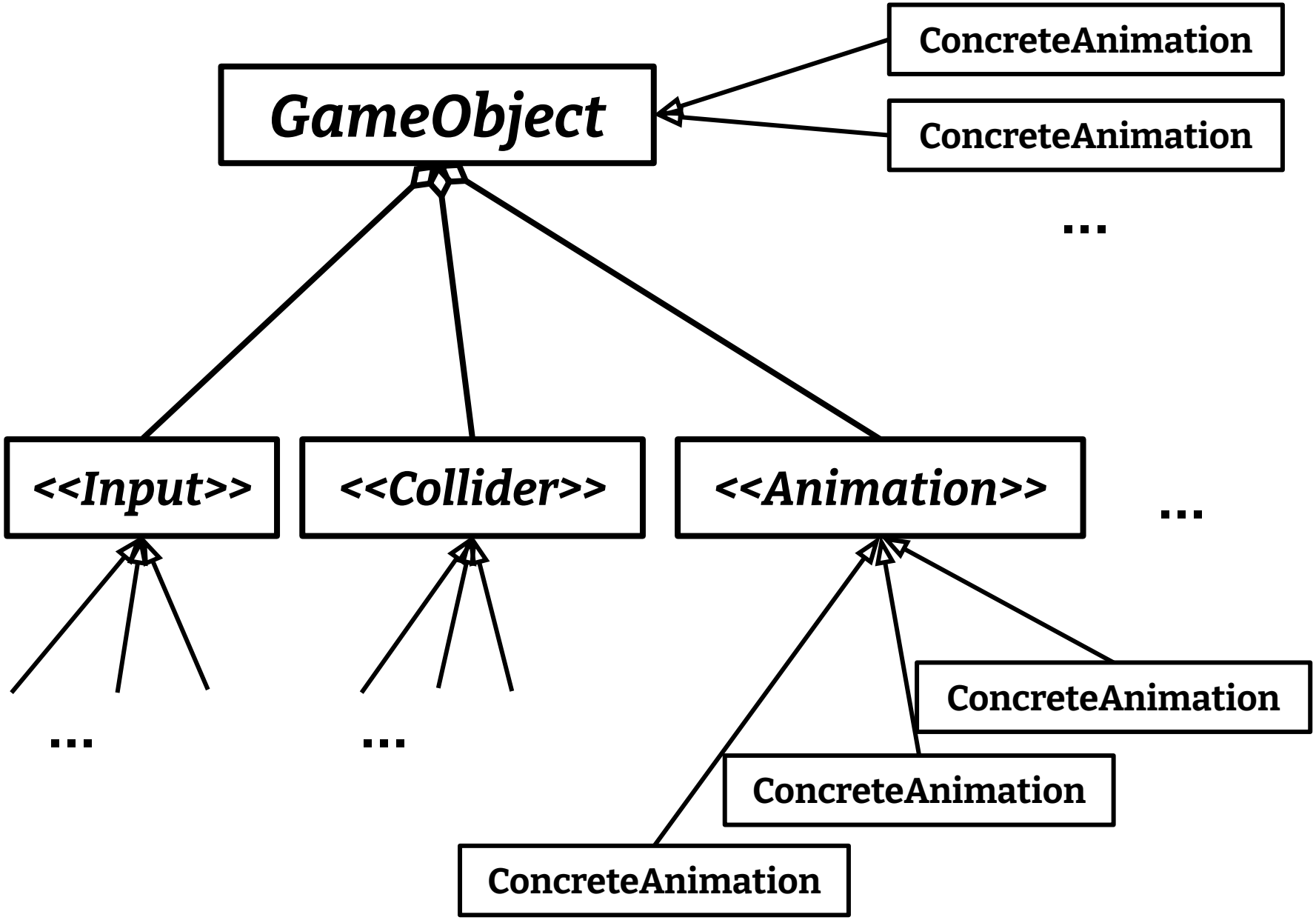

BOOOOOOM!

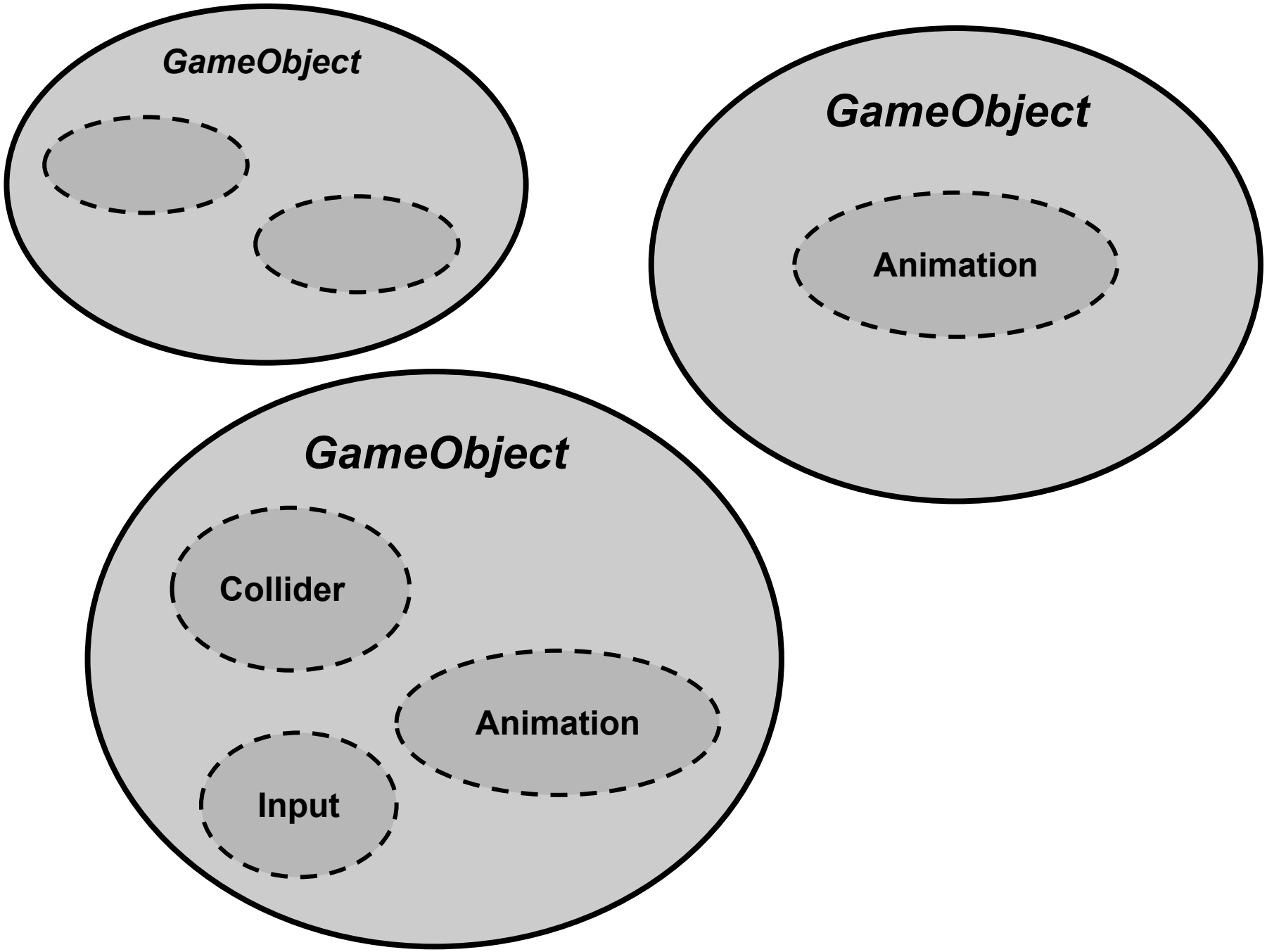
Já existe um *Padrão de Projeto* muito bom para isso:

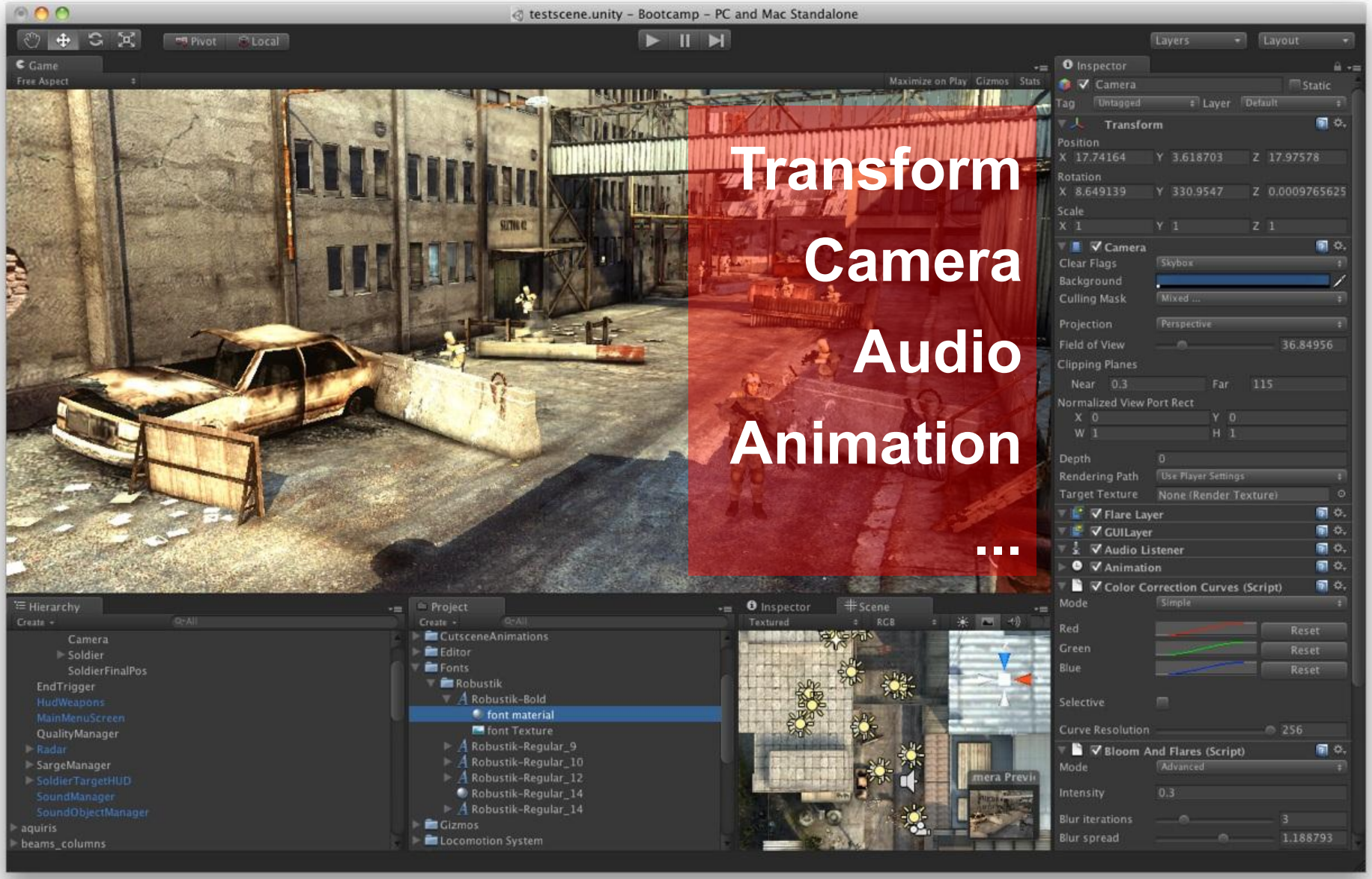
Strategy
(ou Component)

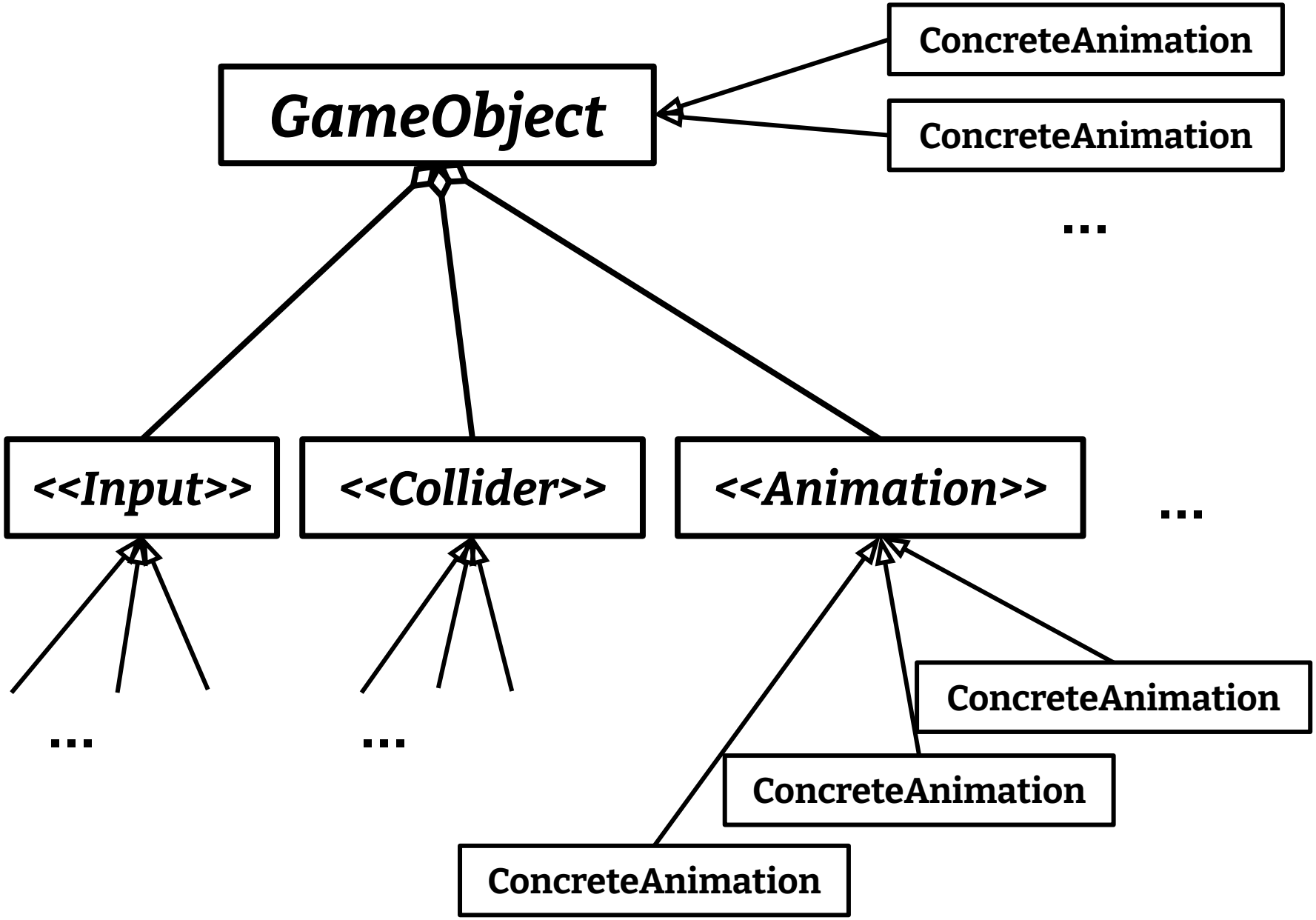
Já existe um *Padrão de Projeto* muito bom para isso:

Strategy
(ou Component)


GameObject
ConcreteAnimation
ConcreteAnimation
...
<<Input>>
<<Collider>>
<<Animation>>
...
...
...
ConcreteAnimation
ConcreteAnimation
ConcreteAnimation

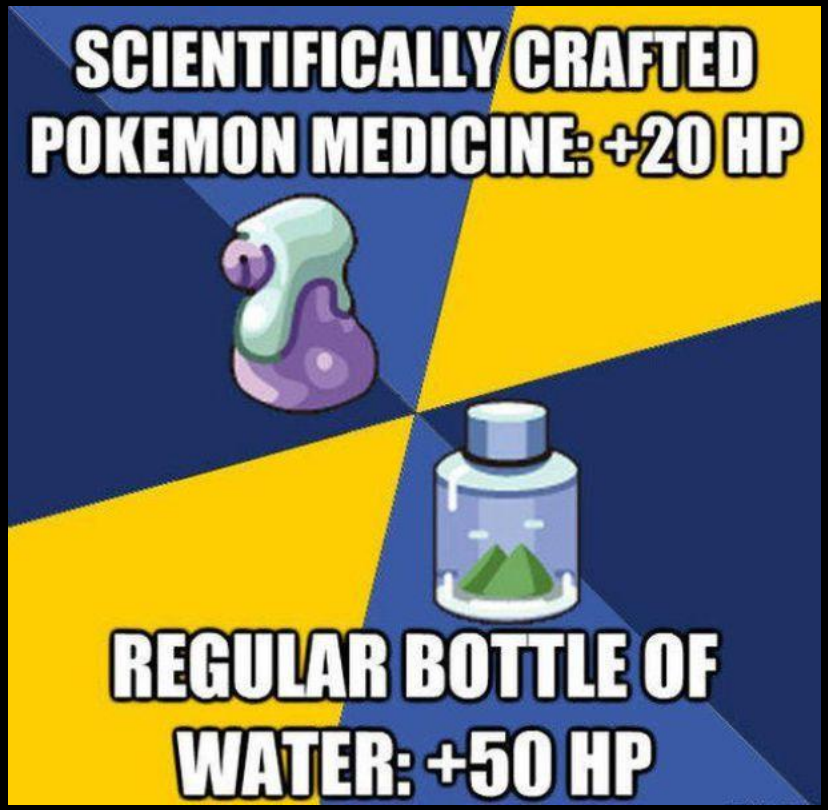

GameObject
GameObject
Animation
GameObject
Collider
Animation
Input


testscene.unity – Bootcamp – PC and Mac Standalone
Transform
Camera
Audio
Animation
...


GameObject
ConcreteAnimation
ConcreteAnimation
...
<<Input>>
<<Collider>>
<<Animation>>
...
...
...
ConcreteAnimation
ConcreteAnimation
ConcreteAnimation

```
class GameObject
{
    public:
        GameObject();

        void handle_input();
        virtual void update() = 0;
        void render();

    private:
        Input* input = NULL;
        Collider* collider = NULL;
        Animation* animation = NULL;
        ...
}
```

```
class Input
{
    public:
        virtual void handle_input() = 0;

    ...
}

class PlayerInput : public Input
{
    public:
        void handle_input();

    ...
}

class EnemyInput : public Input
{
    public:
        void handle_input();

    ...
}
```



Lógica do Jogo

Atualizar

Renderizar

Tratar Inputs


```
//Player 1 logic
player1.handle_collision();
player1.set_status();
player1.move();

//Player 2 logic
player2.handle_collision();
player2.set_status();
player2.move();
```


```
//Player 1 logic
player1.handle_collision();
player1.set_status();
player1.move();

//Player 2 logic
player2.handle_collision();
player2.set_status();
player2.move();
```


```
//Player 1 logic
player1.handle_collision();
player1.set_status();
player1.move();

//Player 2 logic
player2.handle_collision();
player2.set_status();
player2.move();
```


```
//Player 1 logic
player1.handle_collision();
player1.set_status();
player1.move();

//Player 2 logic
player2.handle_collision();
player2.set_status();
player2.move();
```


```
//Player 1 logic
player1.handle_collision();
player1.set_status();
player1.move();

//Player 2 logic
player2.handle_collision();
player2.set_status();
player2.move();
```


```
//Player 1 logic
player1.handle_collision();
player1.set_status();
player1.move();

//Player 2 logic
player2.handle_collision();
player2.set_status();
player2.move();
```


```
//Player 1 logic
player1.handle_collision();
player1.set_status();
player1.move();

//Player 2 logic
player2.handle_collision();
player2.set_status();
player2.move();
```

```
//Handle collision
player1->handle_collision();
player2->handle_collision();

//Set status
player1->set_status();
player2->set_status();

//Move players
player1->move();
player2->move();
```


```
//Handle collision
player1->handle_collision();
player2->handle_collision();

//Set status
player1->set_status();
player2->set_status();

//Move players
player1->move();
player2->move();
```


```
//Handle collision
player1->handle_collision();
player2->handle_collision();

//Set status
player1->set_status();
player2->set_status();

//Move players
player1->move();
player2->move();
```


```
//Handle collision
player1->handle_collision();
player2->handle_collision();

//Set status
player1->set_status();
player2->set_status();

//Move players
player1->move();
player2->move();
```


```
//Handle collision
player1->handle_collision();
player2->handle_collision();

//Set status
player1->set_status();
player2->set_status();

//Move players
player1->move();
player2->move();
```


```
//Handle collision
player1->handle_collision();
player2->handle_collision();

//Set status
player1->set_status();
player2->set_status();

//Move players
player1->move();
player2->move();
```


1P 79000
HI 79000
K.O
54
Blanka
Blanka
ВЫХОД
SUPER
SUPER

Tratar Inputs de todo mundo

Atualizar todo mundo

Renderizar todo mundo

Game Loop

Game Loop

Game Loop

Game Loop

```
while(game_is_running)
{
    handle_input();
    update();
    render();
}
```

FPS

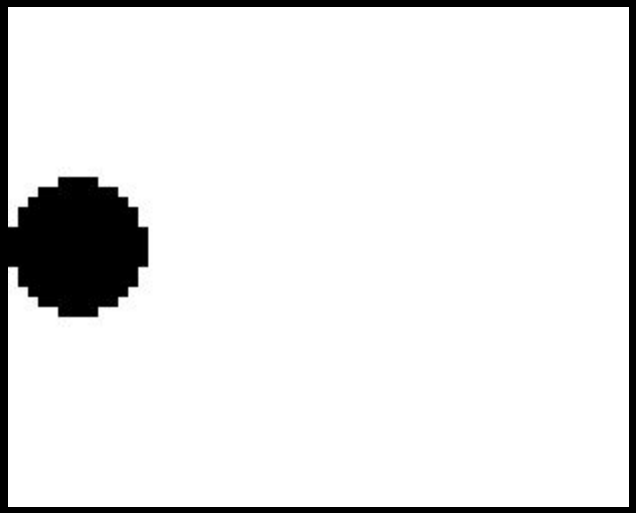
Número de frames renderizados por segundos

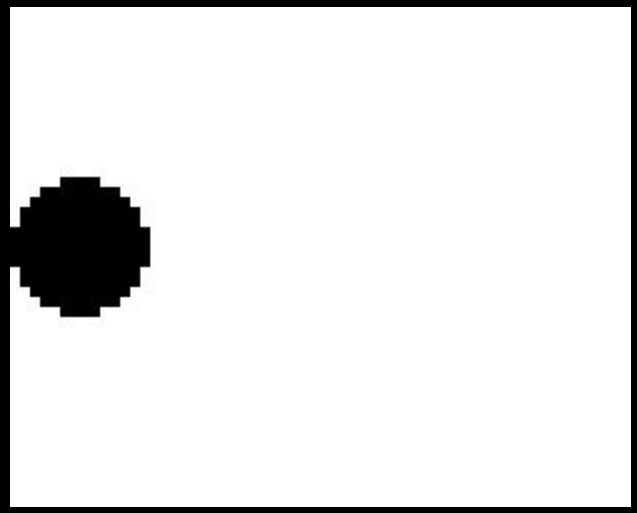
GameSpeed

Número de atualizações do jogo por segundo

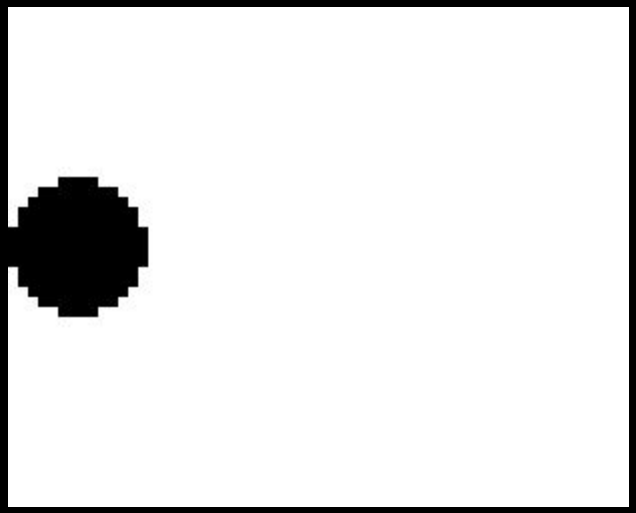
FPS, o olho humano e a taxa de atualização do monitor...

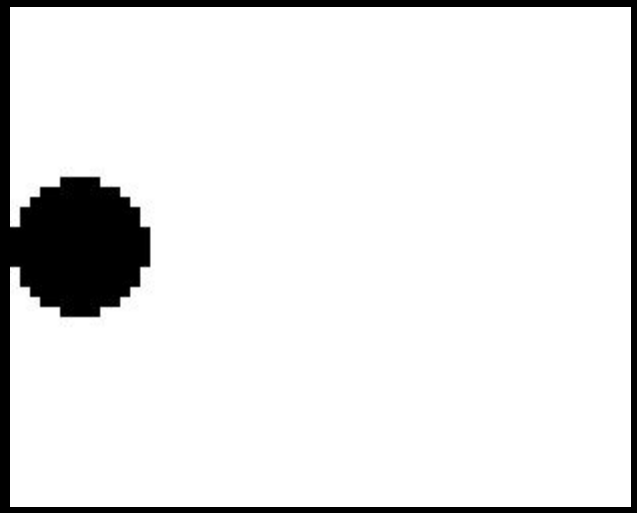
```
while(game_is_running)
{
    handle_input();
    update();
    render();
}
```





```
while(game_is_running)
{
    handle_input();
    update();
    render();

    sleep(until_next_frame);
}
```

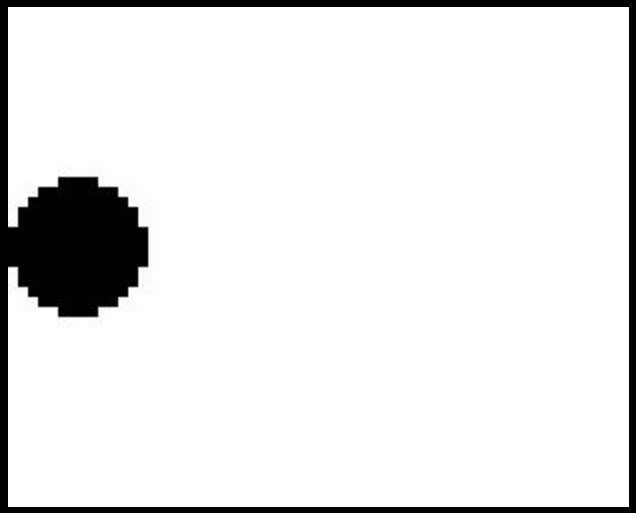


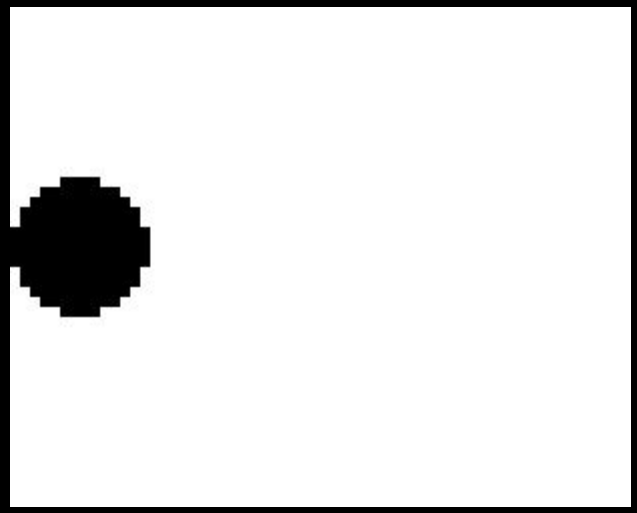


```
prev_frame = 0;
current_frame = get_ticks();

while(game_is_running)
{
    prev_frame = current_frame;
    current_frame = get_ticks();

    handle_input();
    update(current_frame - prev_frame);
    render();
}
```

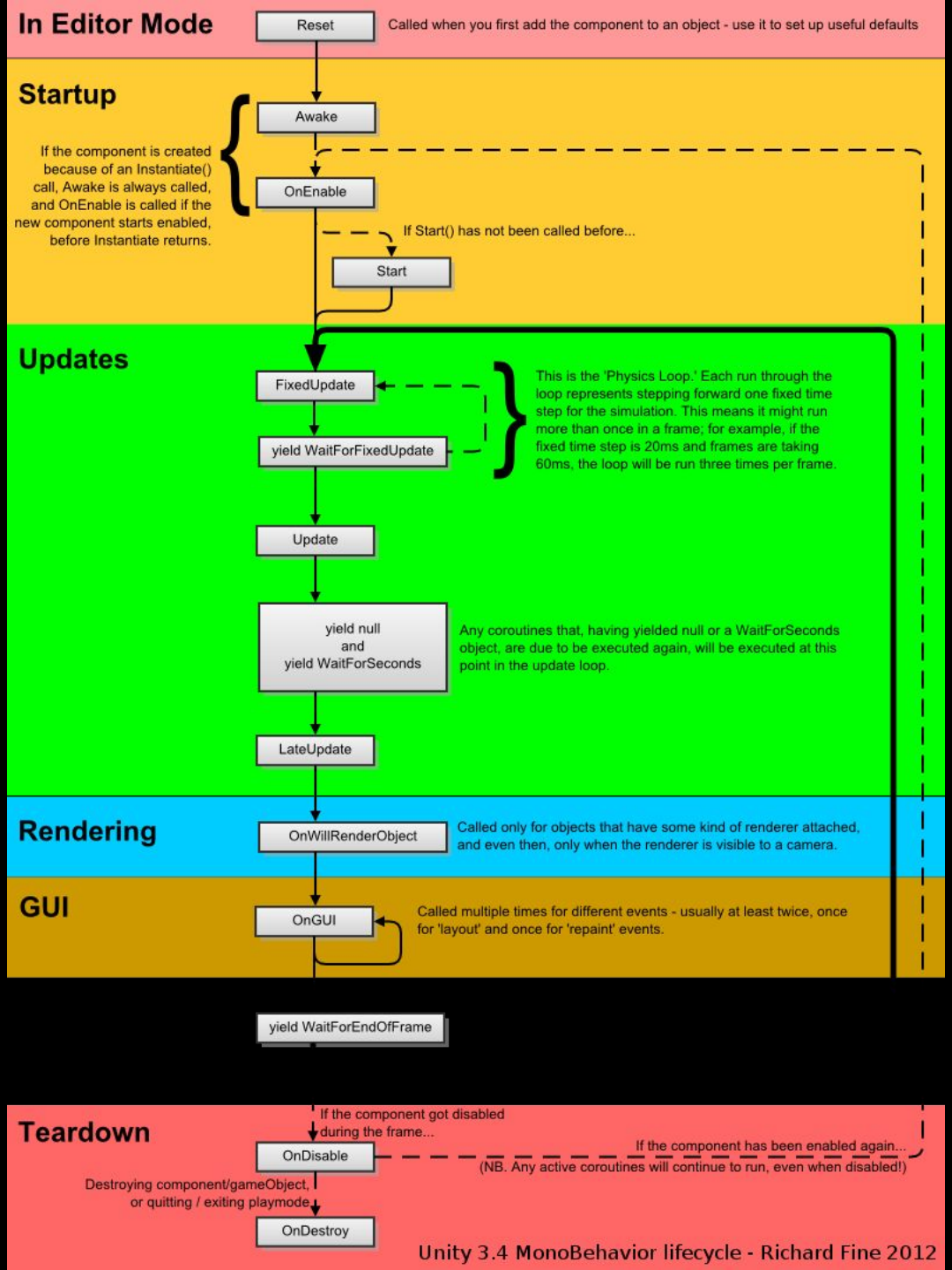
```
lag = 0;
prev_frame = 0;
current_frame = get_ticks();

while(game_is_running)
{
    current_frame = get_ticks();
    elapsed = current_frame - prev_frame;
    prev_frame = current_frame;
    lag += elapsed;

    handle_input();

    while (lag >= MS_PER_UPDATE)
    {
        update();
        lag -= MS_PER_UPDATE;
    }

    render();
}
```

```
lag = 0;
prev_frame = 0;
current_frame = get_ticks();

while(game_is_running)
{
    current_frame = get_ticks();
    elapsed = current_frame - prev_frame;
    prev_frame = current_frame;
    lag += elapsed;

    handle_input();

    while (lag >= MS_PER_UPDATE)
    {
        update();
        lag -= MS_PER_UPDATE;
    }

    render(lag / MS_PER_UPDATE);
}
```





```
lag = 0;
prev_frame = 0;
current_frame = get_ticks();

while(game_is_running)
{
    current_frame = get_ticks();
    elapsed = current_frame - prev_frame;
    prev_frame = current_frame;
    lag += elapsed;

    handle_input();

    while (lag >= MS_PER_UPDATE)
    {
        update();
        lag -= MS_PER_UPDATE;
    }

    render(lag / MS_PER_UPDATE);
}
```

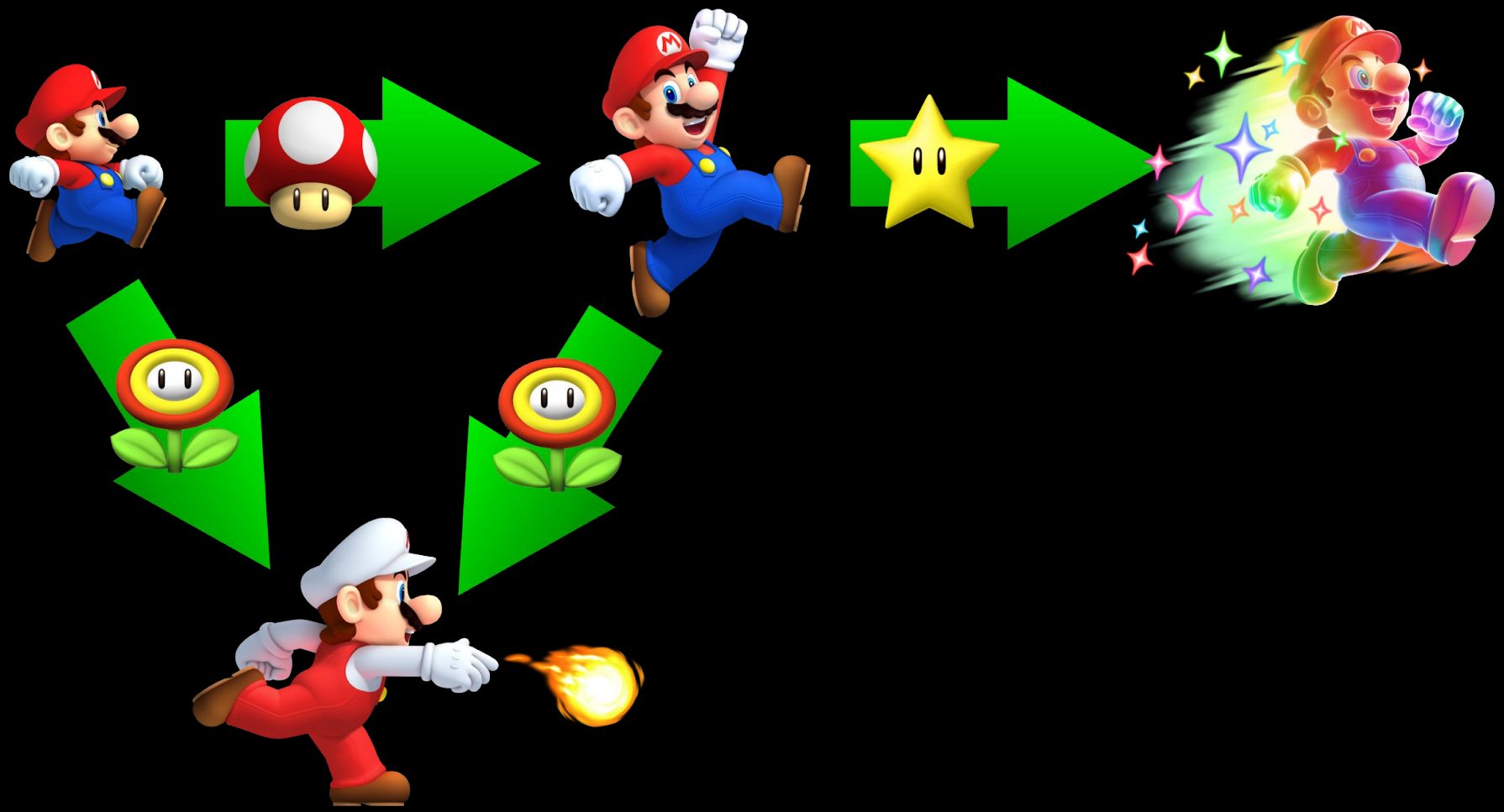

In Editor Mode
Reset
Called when you first add the component to an object - use it to set up useful defaults
Startup
Awake
If the component is created because of an Instantiate() call, Awake is always called, and OnEnable is called if the new component starts enabled, before Instantiate returns.
OnEnable
If Start() has not been called before...
Start
Updates
FixedUpdate
This is the 'Physics Loop.' Each run through the loop represents stepping forward one fixed time step for the simulation. This means it might run more than once in a frame; for example, if the fixed time step is 20ms and frames are taking 60ms, the loop will be run three times per frame.
yield WaitForFixedUpdate
Update
yield null
and
yield WaitForSeconds
Any coroutines that, having yielded null or a WaitForSeconds object, are due to be executed again, will be executed at this point in the update loop.
LateUpdate
Rendering
OnWillRenderObject
Called only for objects that have some kind of renderer attached, and even then, only when the renderer is visible to a camera.
GUI
OnGUI
Called multiple times for different events - usually at least twice, once for 'layout' and once for 'repaint' events.
yield WaitForEndOfFrame
Teardown
If the component got disabled during the frame...
OnDisable
If the component has been enabled again...
(NB. Any active coroutines will continue to run, even when disabled!)
Destroying component/gameObject, or quitting / exiting playmode
OnDestroy
Unity 3.4 MonoBehavior lifecycle - Richard Fine 2012

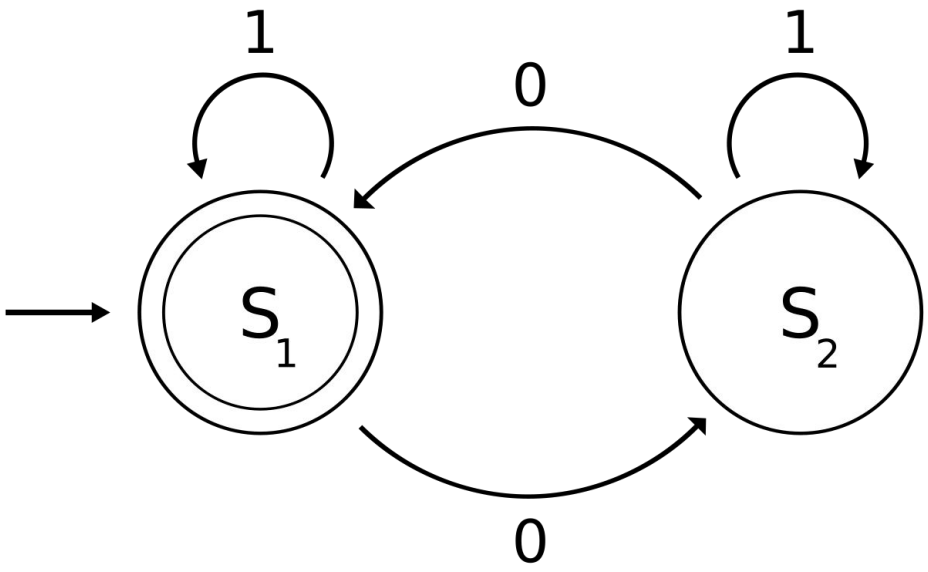
Máquina de Estados Finitos

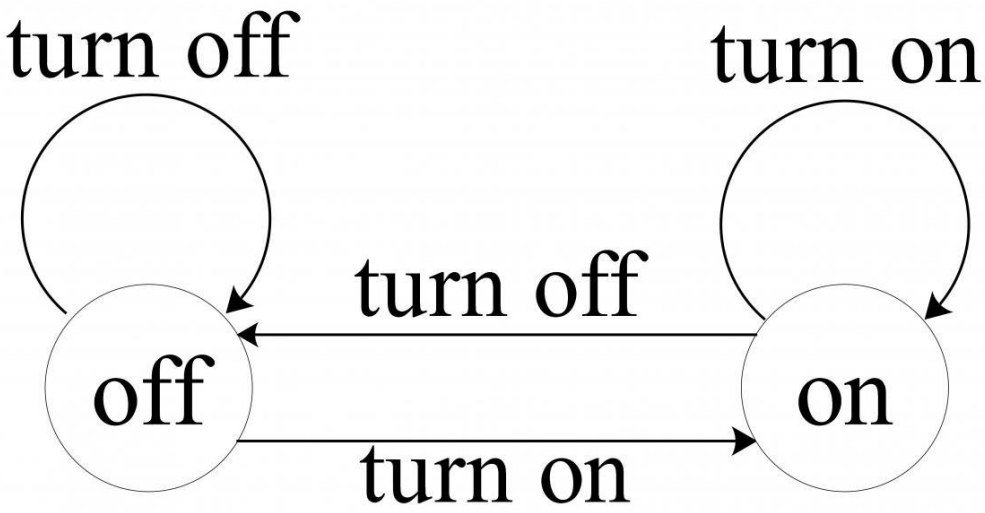
Um simples tratamento de Inputs...

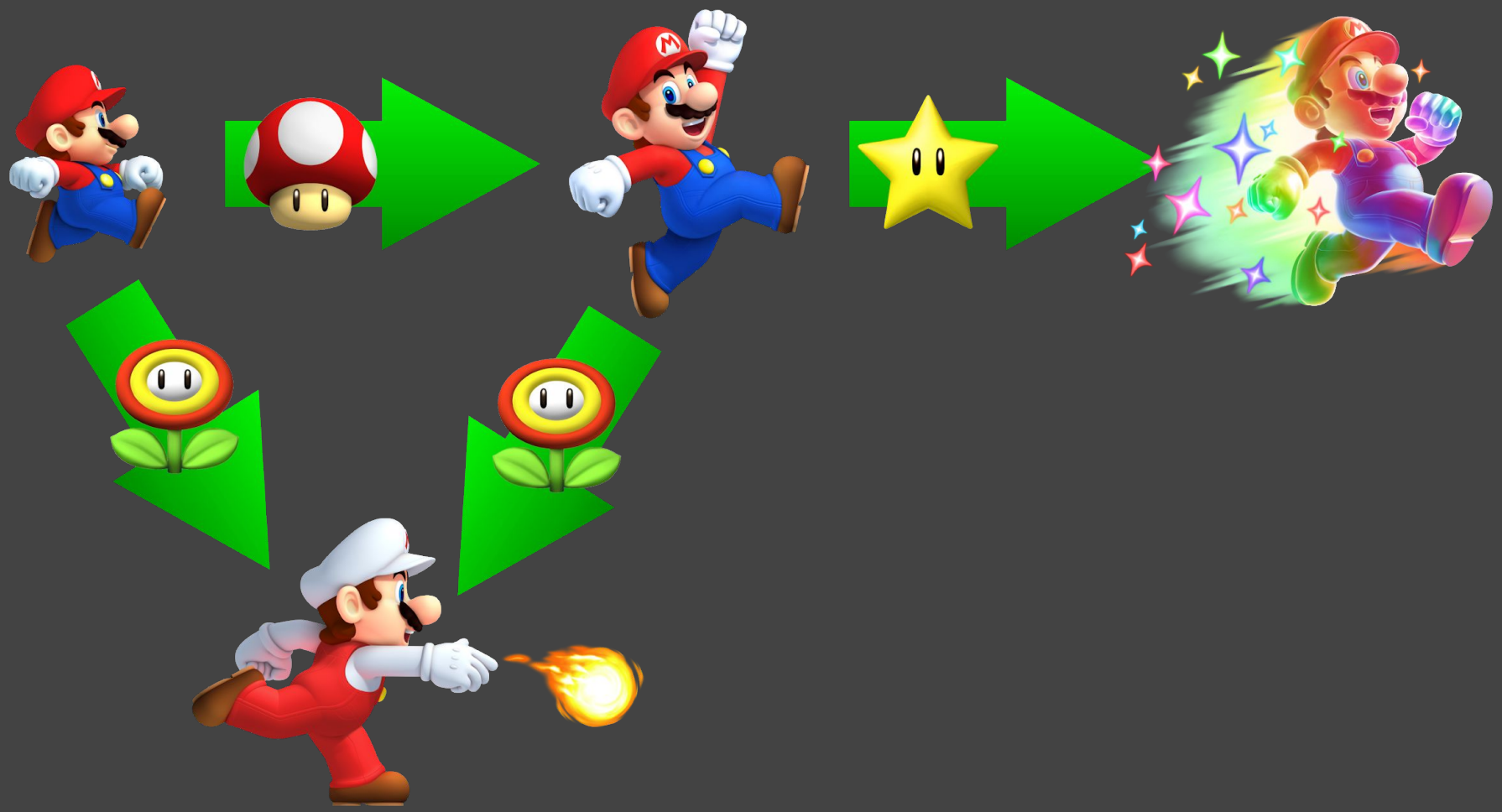
```
void Heroine::handleInput(Input input)
{
    if (input == PRESS_B)
    {
        // Jump if not jumping…
    }
    else if (input == PRESS_DOWN)
    {
        if (!isJumping_)
        {
            setGraphics(IMAGE_DUCK);
        }
    }
    else if (input == RELEASE_DOWN)
    {
        setGraphics(IMAGE_STAND);
    }
}
```

```
void Heroine::handleInput(Input input)
{
    if (input == PRESS_B)
    {
        if (!isJumping_ && !isDucking_)
        {
            // Jump...
        }
    }
    else if (input == PRESS_DOWN)
    {
        if (!isJumping_)
        {
            isDucking_ = true;
            setGraphics(IMAGE_DUCK);
        }
    }
    else if (input == RELEASE_DOWN)
    {
        if (isDucking_)
        {
            isDucking_ = false;
            setGraphics(IMAGE_STAND);
        }
    }
}
```




1
0
1
S1
S2
0


turn off
turn on
turn off
off
on
turn on



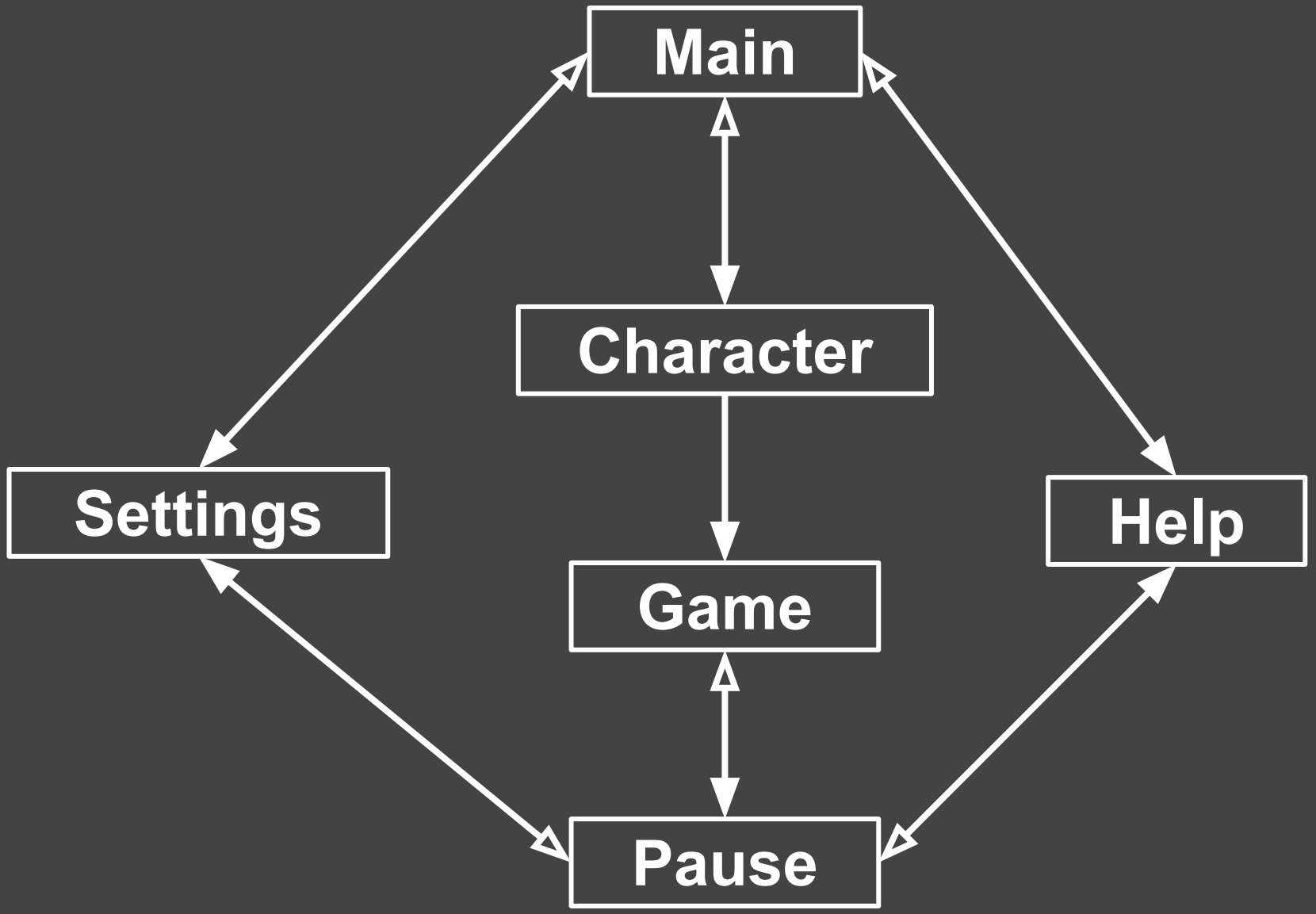
PAUSE

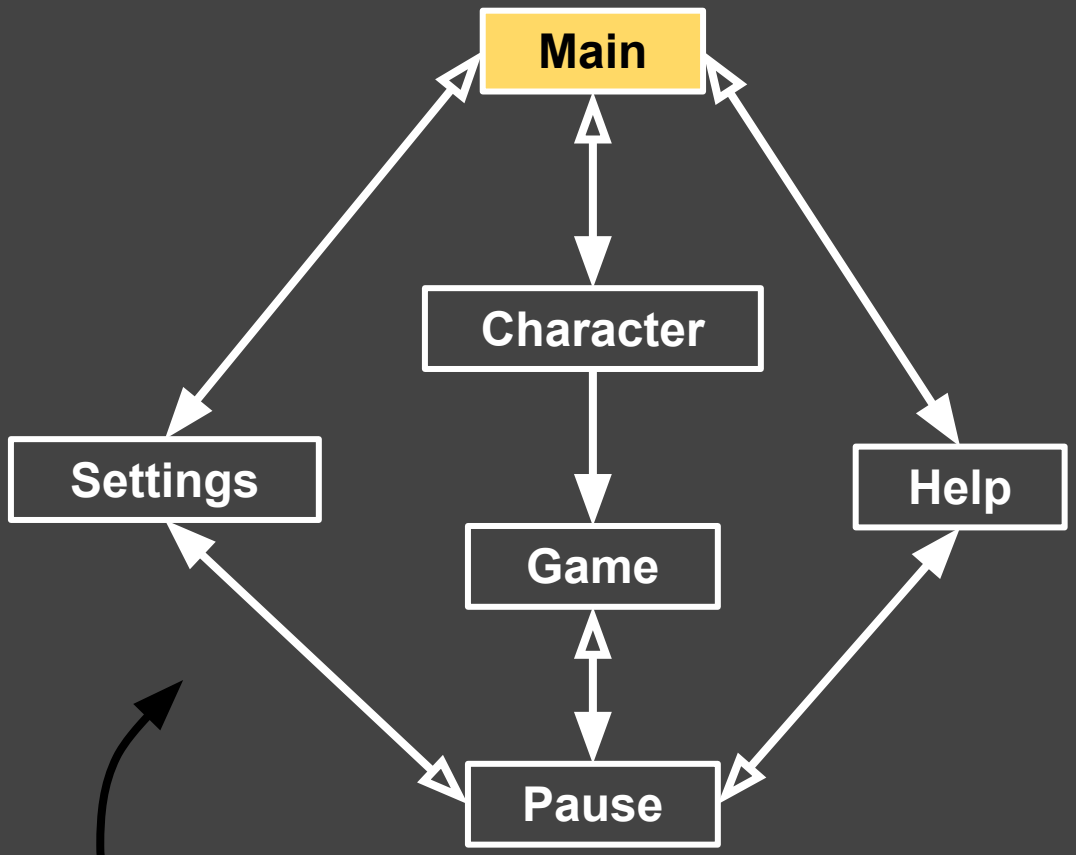
RESUME

SETTINGS

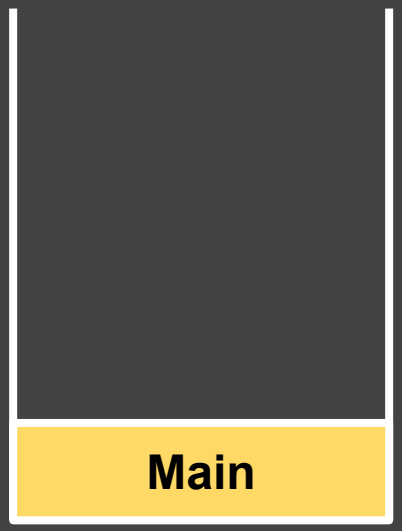
HELP

QUIT

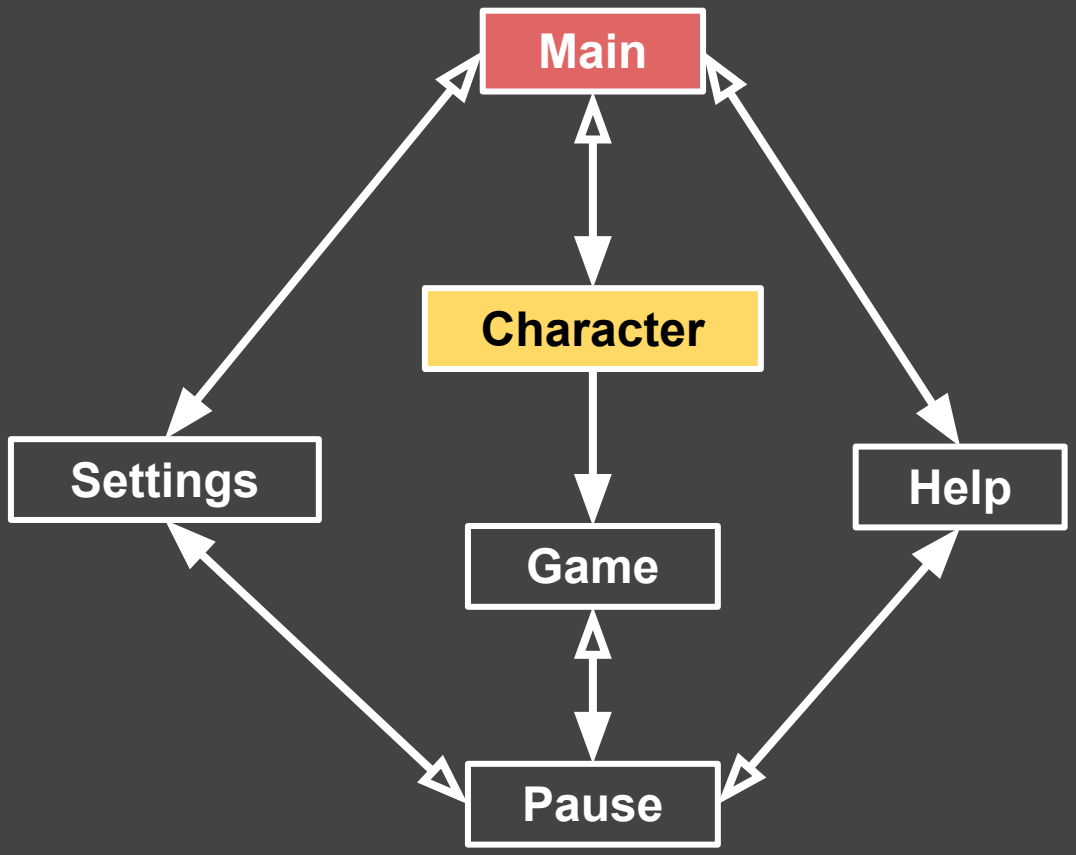

Main
Character
Settings
Help
Game
Pause



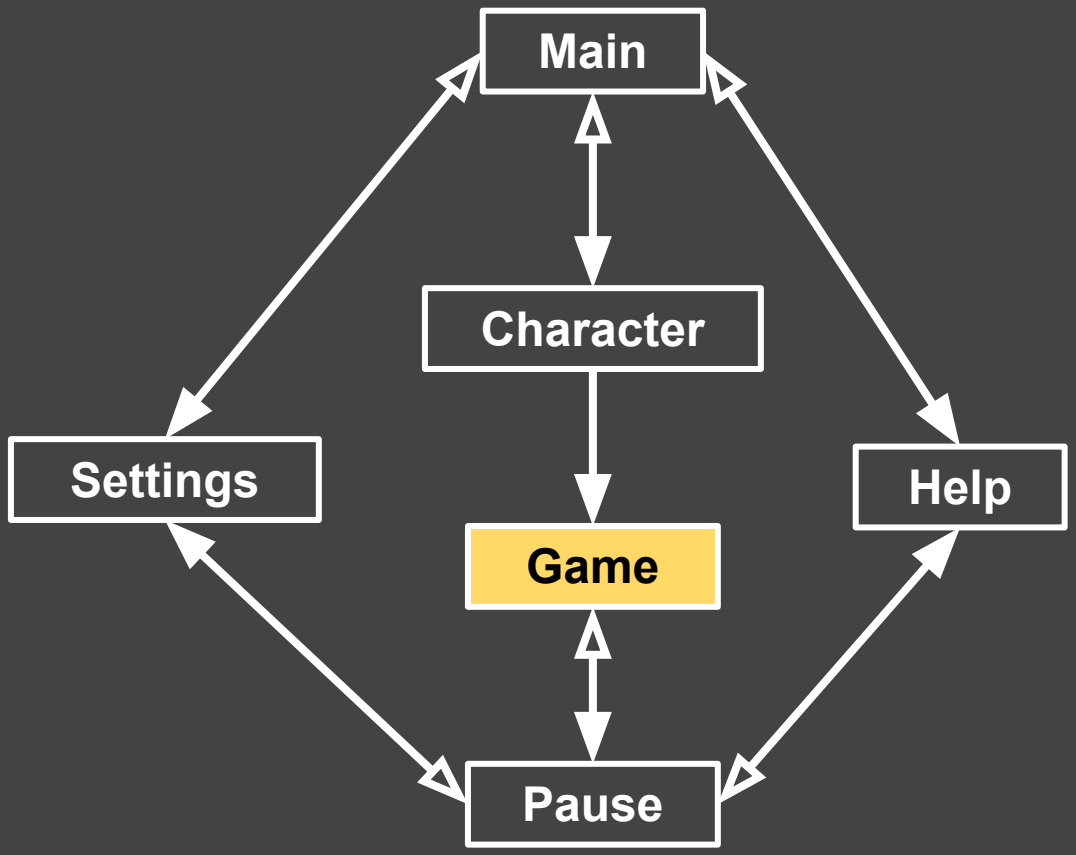
Essa é a FSM

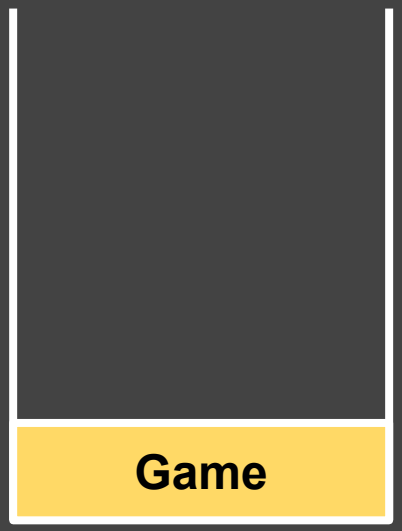


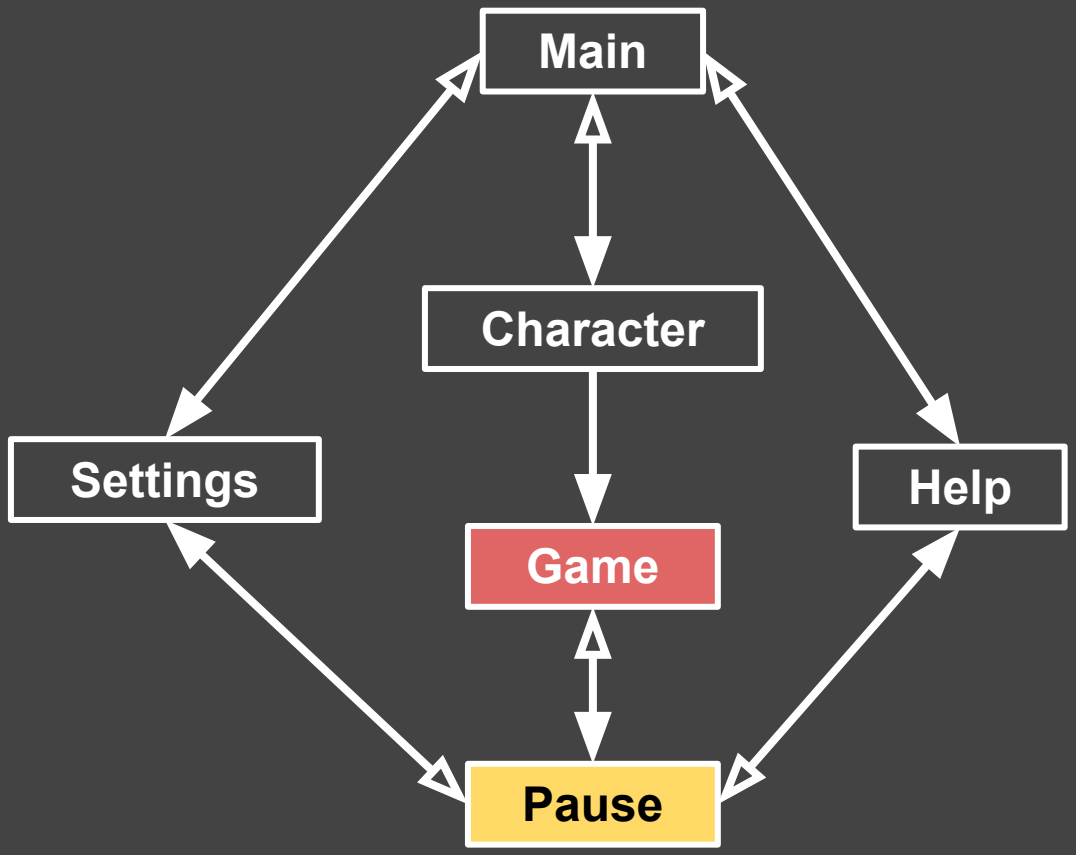
Essa é a pilha

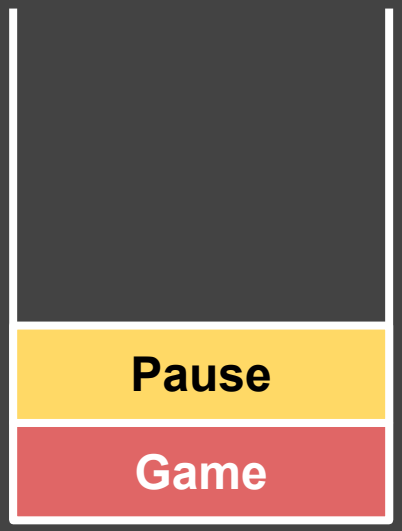

Main
Character
Settings
Help
Game
Pause

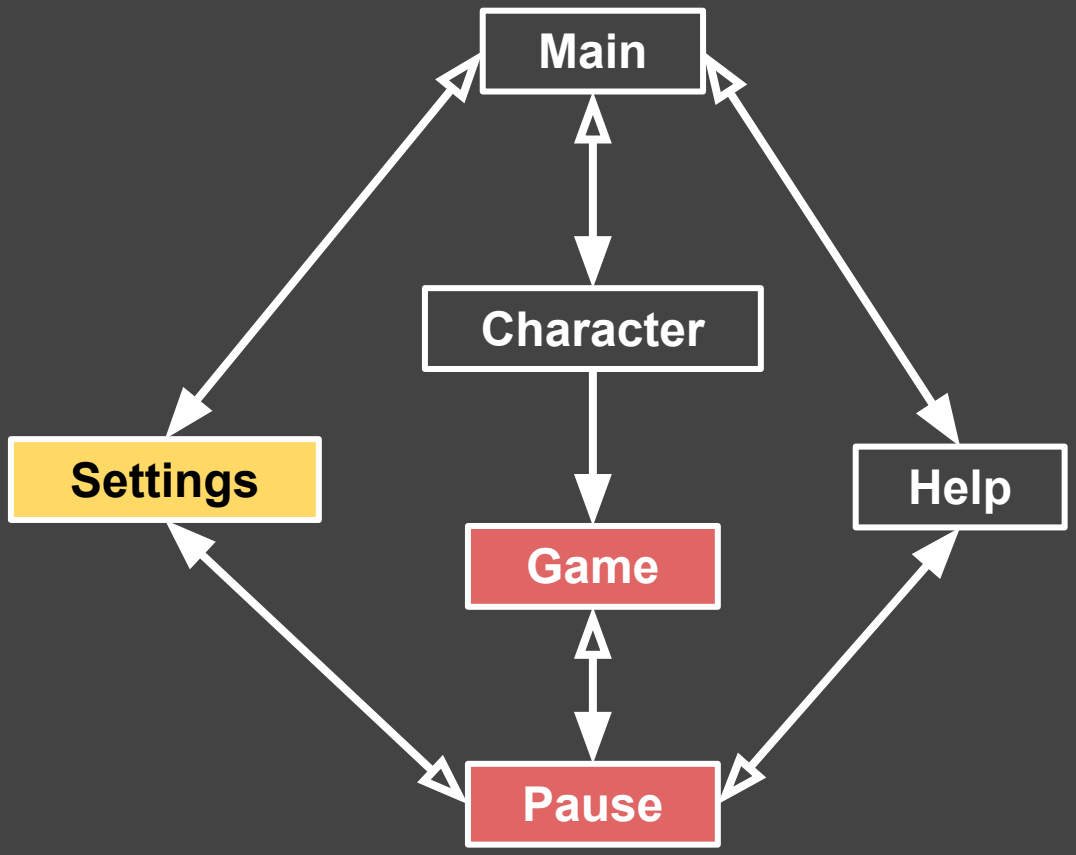

Character
Main

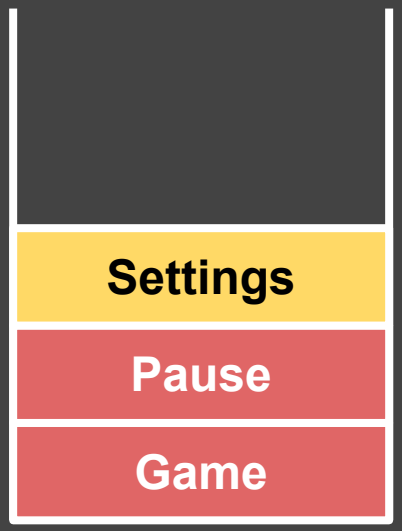

Main
Character
Settings
Help
Game
Pause

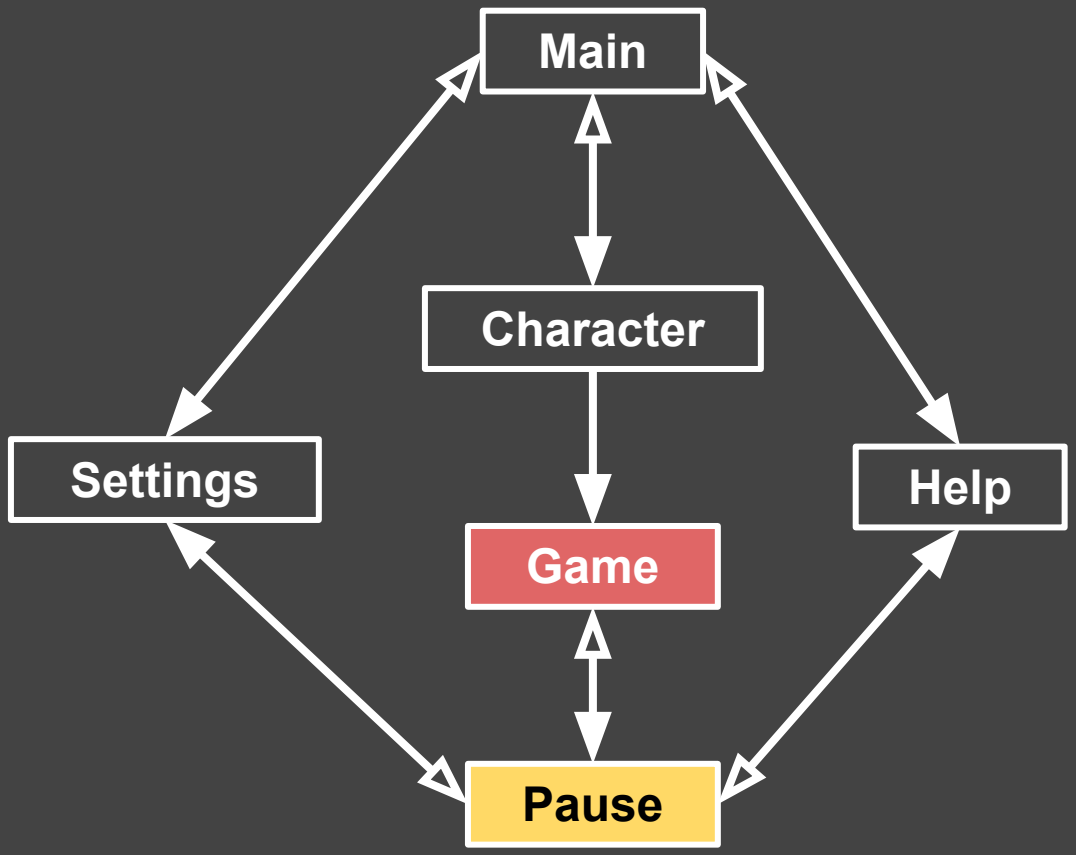

Game

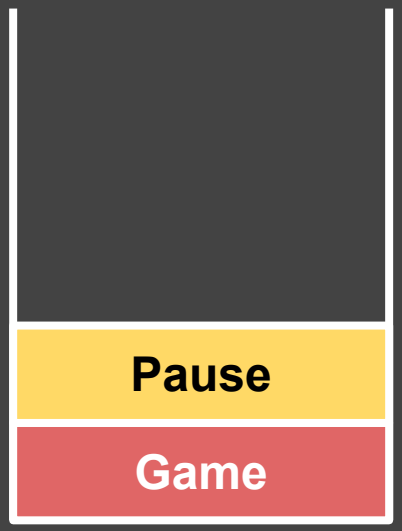

Main
Character
Settings
Help
Game
Pause

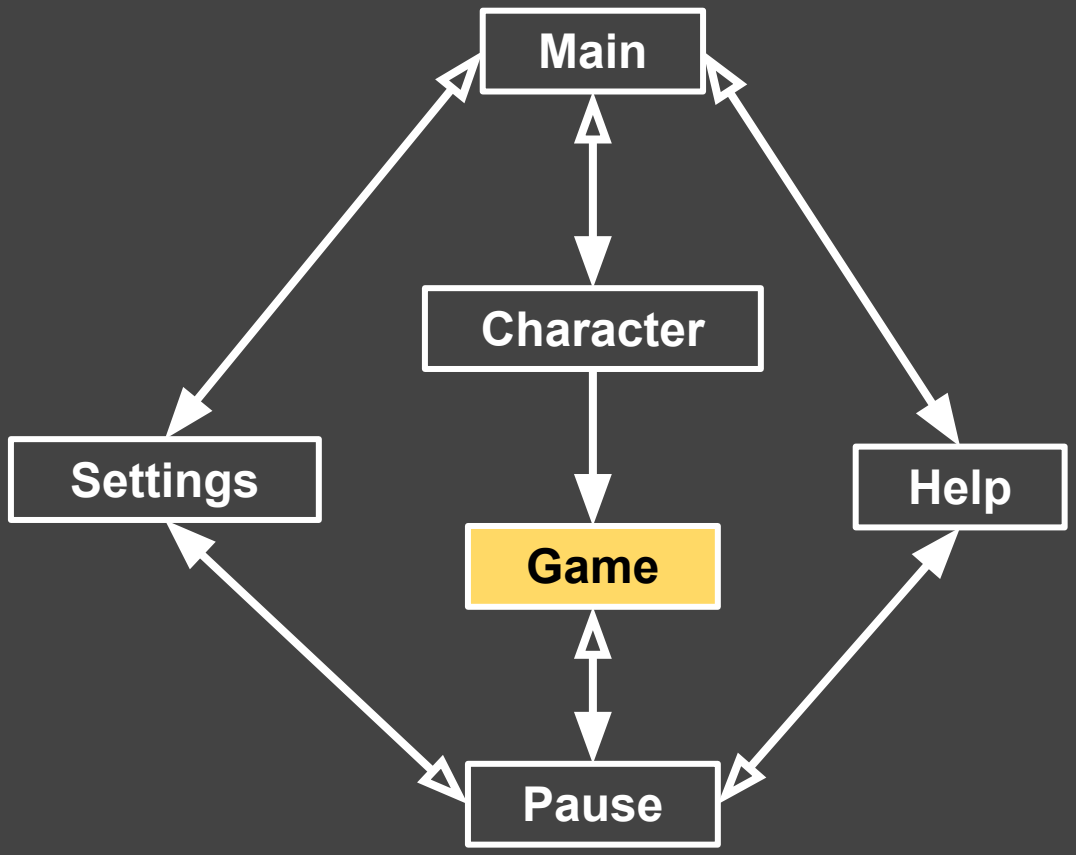

Pause
Game

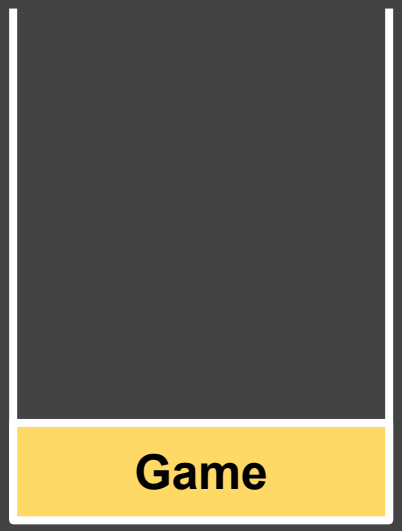

Main
Character
Settings
Help
Game
Pause


Settings
Pause
Game


Main
Character
Settings
Help
Game
Pause


Pause
Game


Main
Character
Settings
Help
Game
Pause


Game


LIFE
RATION


LIFE


ALERT
危険
99.99

Scripting

O que fazer para mudar a posição de um objeto?

Ir no código, mudar o valor e recompilar!

O que fazer para mudar a posição de um objeto?

Ir no código, mudar o valor e recompilar!

// recompilar demora 30 minutos...

Se quiser mudar de mais uma vez?

Ir no código, mudar o valor e recompilar!

Se quiser mudar de mais uma vez?

Ir no código, mudar o valor e recompilar!

// recompilar demora 30 minutos...

Se quiser mudar de mais uma vez?

Ir no código, mudar o valor e recompilar!

Se quiser mudar de mais uma vez?

NÃO!

Ir no código, mudar o valor e recompilar!

Existem maneiras mais inteligentes de fazer isso, ok?

Scripts

```
<GAME height="1280" width="720"/>
    <LEVELONE>
        <tileset="00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                  00 AA AA AA AA EF EF AA AA AA AA AA AA AA 00
                  00 EF 31 31 31 EF 31 31 31 AA 31 31 31 AA 00
                  00 EF 31 EF 2B AA 31 2B 31 AA 31 2B 2B AA 00
                  00 AA 31 31 31 AA 31 31 31 AA 31 AA EF EF 00
                  00 AA 2B EF 31 AA 31 EF 31 AA 31 AA EF EF 00
                  00 AA 31 31 31 AA 31 AA 31 AA 31 31 31 AA 00
                  00 AA 2B 2B EF AA 2B AA EF EF 2B 2B 2B AA 00
                  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00"/>
        <OBJECTS>
            <object texture="player01.png" x="32" y="120" height="32" width="64"/>
            <object texture="enemy01.png" x="240" y="120" height="48" width="55"/>
        </OBJECTS>
    </LEVELONE>
</GAME>
```

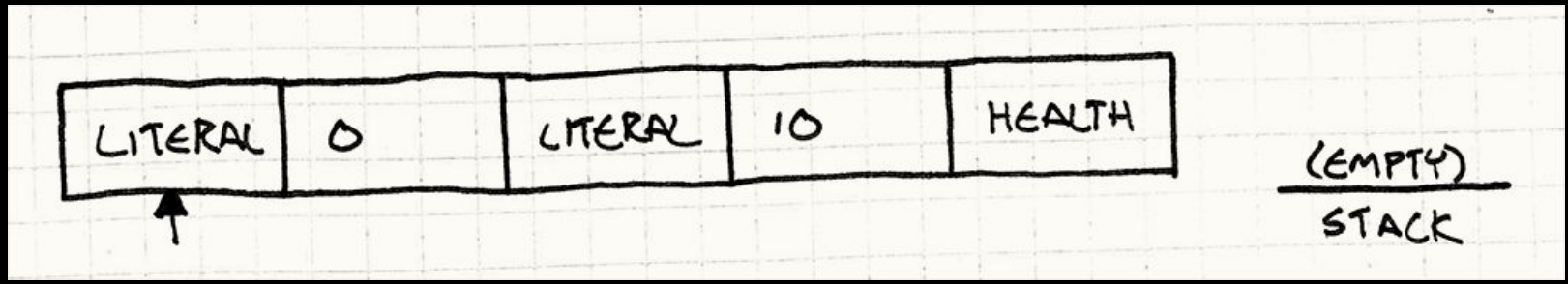
Legal, mas se eu quiser mudar comportamento?

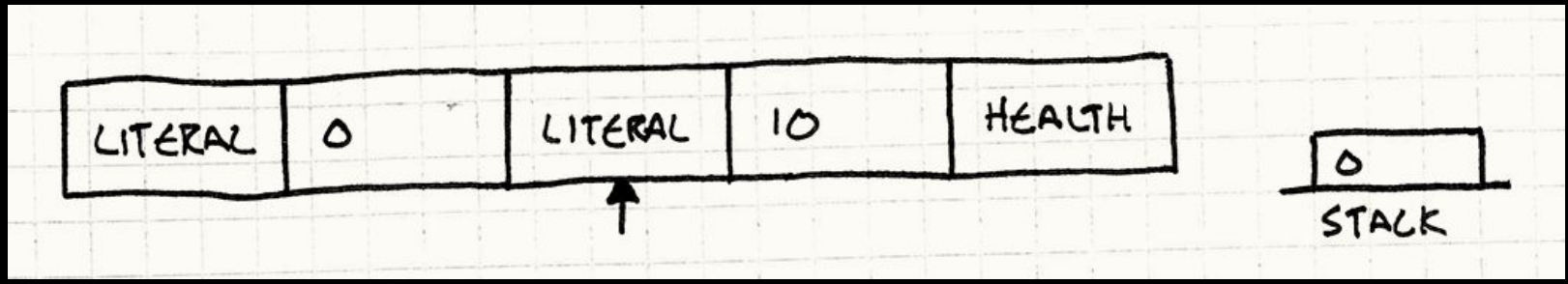
Dividindo a saúde pela metade:

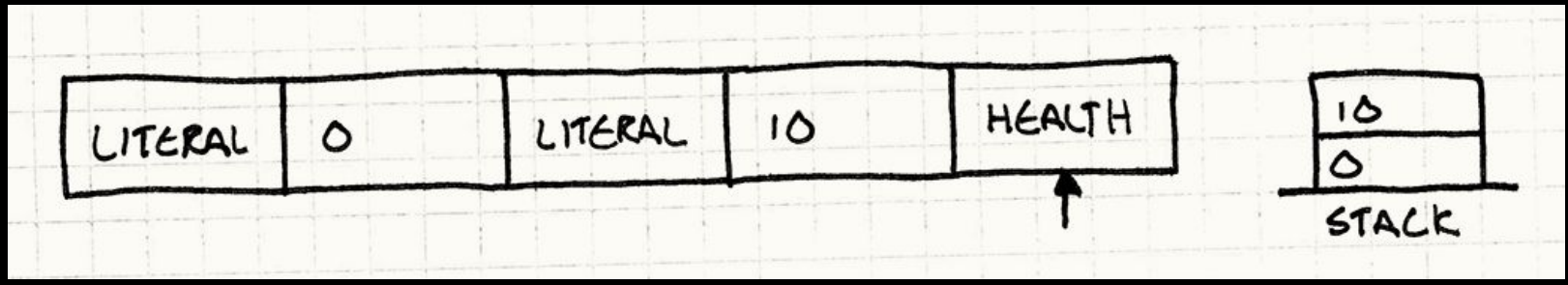
```
set_health( get_health() / 2 );
```

Dividindo a saúde pela metade:

```
set_health( get_health() / 2 );
```







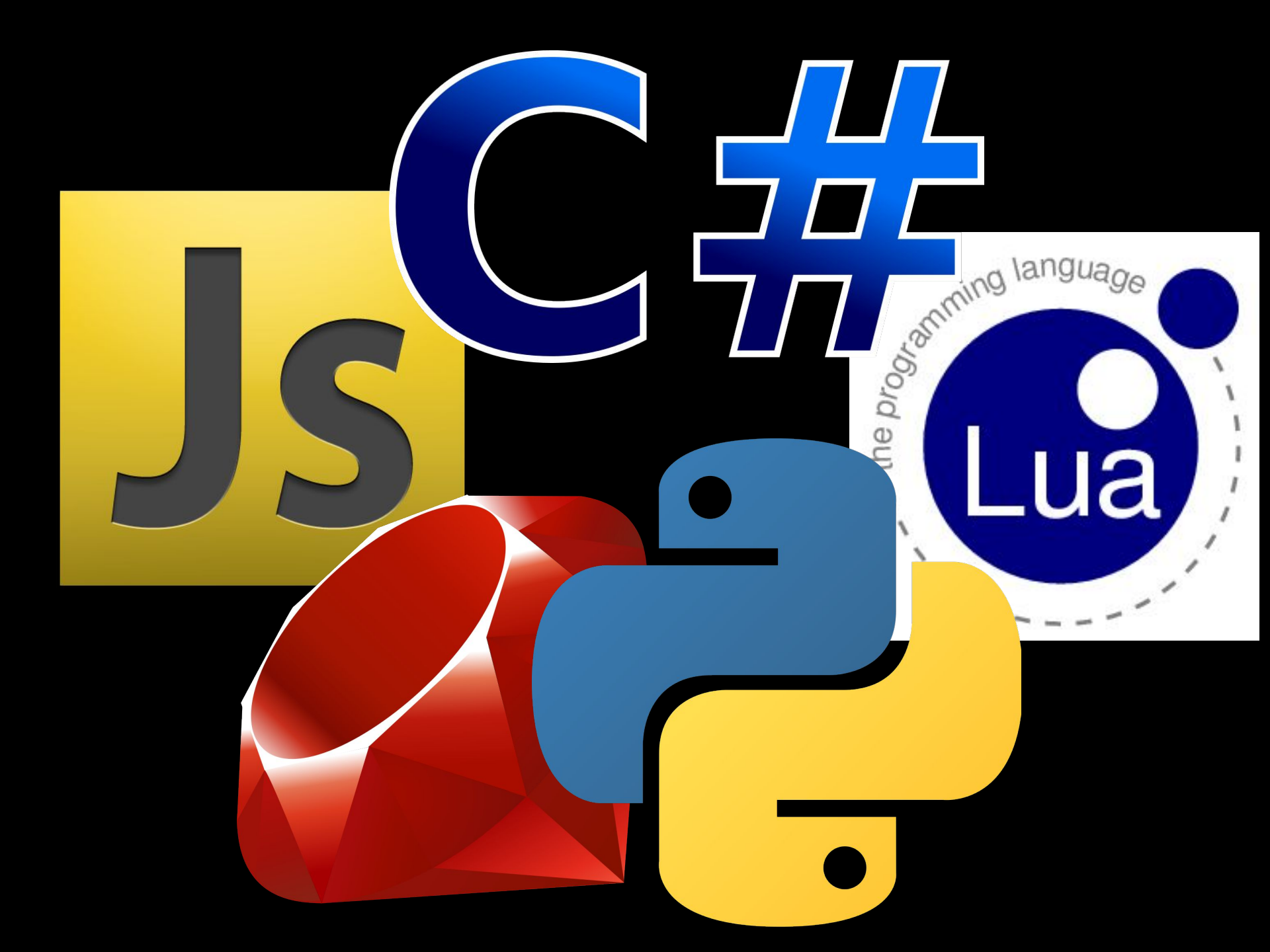
```
LITERAL 0    [0]            # Wizard index
LITERAL 0    [0, 0]         # Wizard index
GET_HEALTH   [0, 45]        # getHealth()
LITERAL 0    [0, 45, 0]     # Wizard index
GET_AGILITY  [0, 45, 7]     # getAgility()
LITERAL 0    [0, 45, 7, 0]  # Wizard index
GET_WISDOM   [0, 45, 7, 11] # getWisdom()
ADD          [0, 45, 18]    # Add agility and wisdom
LITERAL 2    [0, 45, 18, 2] # Divisor
DIVIDE       [0, 45, 9]     # Average agility and wisdom
ADD          [0, 54]        # Add average to current health
SET_HEALTH   []             # Set health to result
```

```
enum Instruction
{
    INST_SET_HEALTH      = 0x00,
    INST_SET_WISDOM      = 0x01,
    INST_SET_AGILITY     = 0x02,
    INST_PLAY_SOUND      = 0x03,
    INST_SPAWN_PARTICLES = 0x04
};
```

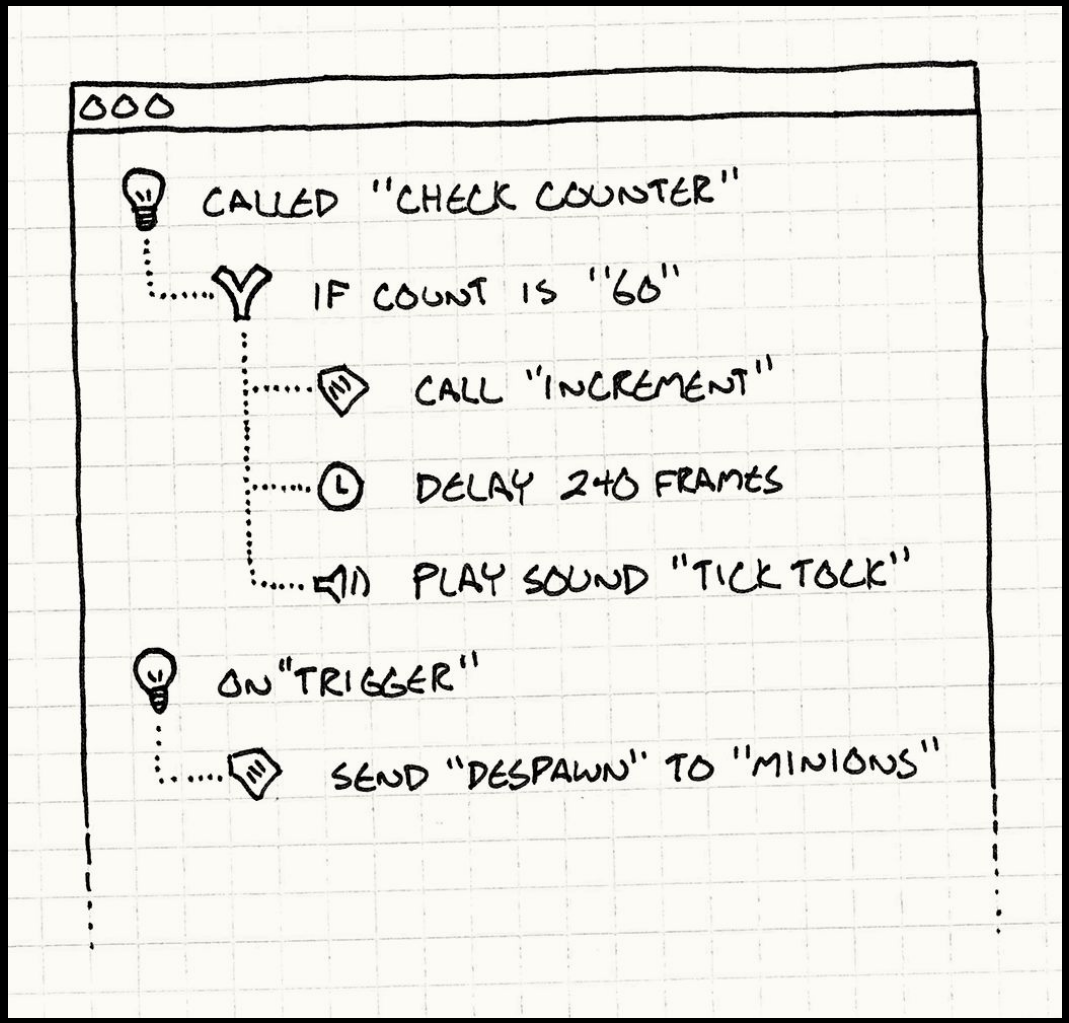
```
switch (instruction)
{
    case INST_SET_HEALTH:
    {
        int amount = pop();
        int wizard = pop();
        setHealth(wizard, amount);
        break;
    }

    case INST_SET_WISDOM:
    case INST_SET_AGILITY:
    // Same as above...

    case INST_PLAY_SOUND:
        playSound(pop());
        break;

    case INST_SPAWN_PARTICLES:
        spawnParticles(pop());
        break;
}
```

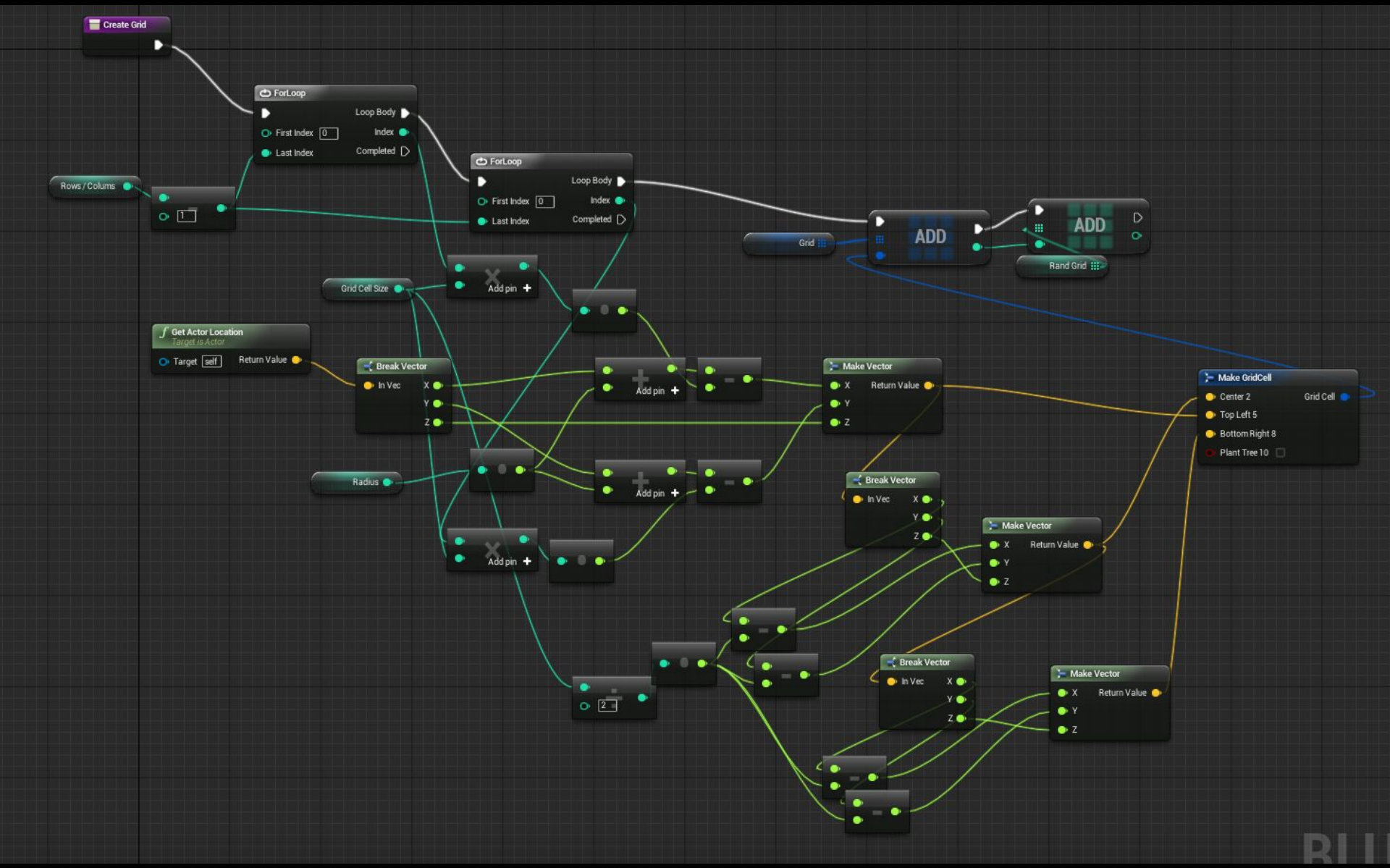
```
LITERAL 0    [0]            # Wizard index
LITERAL 0    [0, 0]         # Wizard index
GET_HEALTH   [0, 45]        # getHealth()
LITERAL 0    [0, 45, 0]     # Wizard index
GET_AGILITY  [0, 45, 7]     # getAgility()
LITERAL 0    [0, 45, 7, 0]  # Wizard index
GET_WISDOM   [0, 45, 7, 11] # getWisdom()
ADD          [0, 45, 18]    # Add agility and wisdom
LITERAL 2    [0, 45, 18, 2] # Divisor
DIVIDE       [0, 45, 9]     # Average agility and wisdom
ADD          [0, 54]        # Add average to current health
SET_HEALTH   []             # Set health to result
```

Scripting languages!!!


JS
C#
the programming language
Lua

```
LITERAL 0    [0]            # Wizard index
LITERAL 0    [0, 0]         # Wizard index
GET_HEALTH   [0, 45]        # getHealth()
LITERAL 0    [0, 45, 0]     # Wizard index
GET_AGILITY  [0, 45, 7]     # getAgility()
LITERAL 0    [0, 45, 7, 0]  # Wizard index
GET_WISDOM   [0, 45, 7, 11] # getWisdom()
ADD          [0, 45, 18]    # Add agility and wisdom
LITERAL 2    [0, 45, 18, 2] # Divisor
DIVIDE       [0, 45, 9]     # Average agility and wisdom
ADD          [0, 54]        # Add average to current health
SET_HEALTH   []             # Set health to result
```




Create Grid
ForLoop
Loop Body
First Index
Index
Last Index
Completed
ForLoop
Loop Body
First Index
Index
Last Index
Completed
Rows / Colums
Grid
ADD
ADD
Rand Grid
Grid Cell Size
Add pin
Get Actor Location
Target is Actor
Target
self
Return Value
Break Vector
In Vec
X
Y
Z
Add pin
Make Vector
X
Y
Z
Return Value
Make GridCell
Center 2
Grid Cell
Top Left 5
Bottom Right 8
Plant Tree 10
Radius
Add pin
Break Vector
In Vec
Make Vector
Return Value
Add pin
2
Break Vector
In Vec
Make Vector
Return Value

Jogos são softwares.

Esteja atento a todas as técnicas de Desenvolvimento de Softwares e use-as!

Jogos são softwares.

Esteja atento a todas as técnicas de Desenvolvimento de Softwares e use-as!

Alguns Padrões de Projetos que podem ser usados:

Strategy

State

Command

Decorator

...

Alguns Padrões de Projetos que podem ser usados:

Strategy

State

Command

Decorator

...

Dicas de performance

Ponteiros

Strings

Evitar containers

Definições em tempo de execução

Classes abstratas, funções virtuais…

Cuidado com a alocação de memória

Cuidado!

Ponteiros

Strings

Evitar containers

Definições em tempo de execução

Classes abstratas, funções virtuais…

Cuidado com a alocação de memória

Cuidado!

Ponteiros

Strings

Evitar containers

Definições em tempo de execução

Classes abstratas, funções virtuais…

Cuidado com a alocação de memória

Cuidado!

Ponteiros

Strings

Evitar containers

Definições em tempo de execução

Classes abstratas, funções virtuais…

Cuidado com a alocação de memória

Cuidado!

Ponteiros

Strings

Evitar containers

Definições em tempo de execução

Classes abstratas, funções virtuais…

Cuidado com a alocação de memória

Cuidado!

Ponteiros

Strings

Evitar containers

Definições em tempo de execução

Classes abstratas, funções virtuais…

Cuidado com a alocação de memória

Cuidado!

Ponteiros

Strings

Evitar containers

Definições em tempo de execução

Classes abstratas, funções virtuais…

Cuidado com a alocação de memória

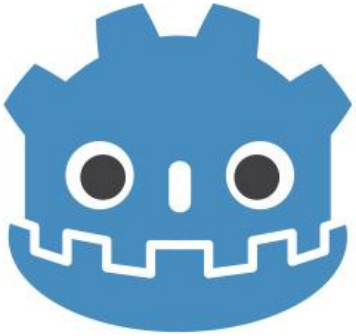
Cuidado!

Engines

UNREAL
ENGINE

CRYENGINE® 3

unity



GODOT
Game engine

GameMaker
Studio

Corona SDK

RPG MAKER® XP

HAXE

Bibliotecas

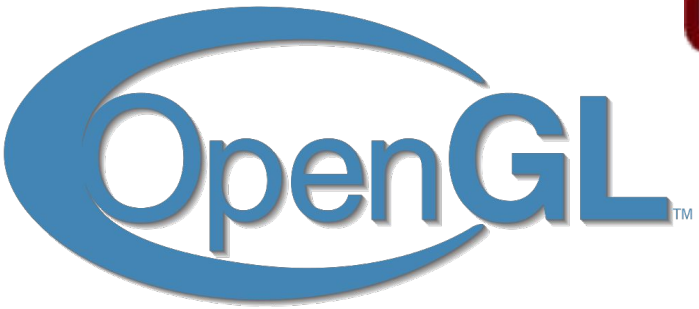

SDL
Simple Directmedia Layer


TINY
XML


Box2D


the FreeType Project


OpenAL


OpenGL


zlib

Sites, Tutoriais e Livros

Sites / Tutoriais

lazyfoo.net

sdltutorials.com

gameprogrammingpatterns.com

Livros

SDL Game Development

Shaun Mitchell

Programming Game AI by Example

Mat Buckland

Software Engineering for Game Developers

John Flynt

Obrigado!

Paulo Bruno de Sousa Serafim

Arthur Carvalho Walraven