Game Design Document (GDD)

None

None

None

Table of contents

1. GAME DESIGN DOCUMENT

Template based on Unity's GDD Template. Available at: Unity's GDD Template

1.1 Game Name:

1.1.1 Genre:

Genre

1.1.2 Game Elements:

Game elements are the basic activities the player will be doing for fun

1.1.3 Player:

The number of players that can play the game at once

1.2 TECHNICAL SPECS

1.2.1 Technical Form:

Basically there are 2D graphics (flat) and 3D graphics (form)

1.2.2 View:

Camera view from which the player will experience the game

1.2.3 Platform:

iOS, Android, Mac, PC, Console...

1.2.4 Language:

C#, C++, Ruby, Java

1.2.5 Device:

PC, Mobile, Console

1.3 GAME PLAY

Use the game play section to create a descriptive paragraph about how the game is played. You want the use tor imagine they are actually playing the game. Try not to use generic (i.e., broad, non-descriptive) terms when writing about the game play. For example, few readers want to hear statements such as, “enemy_1 will have more hit points than enemy_2.” Instead, it’s better to make statements like, “The Lazarus Fighter has more armour than the Apollo Fighter.”

1.3.1 Game Play Outline

This outline will vary depending on the type of game.

- Opening the game application
- Game options
- Story synopsis
- Modes
- Game elements
- Game levels
- Player’s controls
- Winning
- Losing
- End
- Why is all this fun?

1.3.2 Key Features

Key features are a list of game elements that are attractive to the player.

1.4 DESIGN DOCUMENT

This document describes how GameObjects behave, how they’re controlled and their properties. This is often referred to as the “mechanics” of the game. This documentation is primarily concerned with the game itself. This part of the document is meant to be modular, meaning you could have several different Game Design Documents attached to the Concept Document.

1.4.1 Design Guidelines

This is an important statement about any creative restrictions that need to be considered and includes brief statements about the general (i.e., overall) goal of the design.

1.4.2 Game Design Definitions

This section established the definition of the game play. Definitions should include how a player wins, loses, transitions between levels, and the main focus of the gameplay.

1.4.3 Game Flowchart

The game flowchart provides a visual of how the different game elements and their properties interact. Game flowcharts should represent Objects, Properties, and Actions present in the game. Each of these items should have a number reference to where they exist within the game mechanics document.

- Menu
- Synopsis
- Game Play
- Player Control
- Game Over (Winning and Losing)

1.4.4 Player Definition

- Use this section for quick descriptions that define the player

- Use the Player Properties section (below) to define the properties for each player. Player Properties can be affected by the player's action or interaction with other game elements. Define the properties and how they affect the player's current game.
- Use the Player Rewards section to make a list of all objects that affect the player in a positive way. Define these objects by describing what affect they cause and how the player can use the object.

Player Definitions

A suggested list may include:

- Health
- Weapons
- Actions

Player Properties

Each property should mention a feedback as a result of the property changing.

Player Rewards (power-ups and pick-ups)

Make a list of all objects that affect the player in a positive way (e.g., health replenished).

1.4.5 User Interface (UI)

This is where you'll include a description of the user's control of the game. Think about which buttons on a device would be best suited for the game. Consider what the worst layout is, then ask yourself if your UI is it still playable. A visual representation can be added where you relate the physical controls to the actions in the game. When designing the UI, it may be valuable to research quality control and user interface (UI) design information.

2. Como editar o documento

O mkdocs utiliza a sintaxe markdown para criação e edição das páginas. Para criar uma nova página basta criar um arquivo, através do Visual Studio Code, com a extensão md (botão direito na pasta da estrutura de arquivos, new, file e dê o nome e coloque a extensão .md).

Para saber mais sobre a formatação markdown, clique aqui.

Para consultar a documentação oficial do mkdocs, clique aqui

2.1 Título

Os títulos são escritos com o cerquilho (#) sendo que:

```
# para título 1
## para título 2
### para título 3
#### para título 4
##### para título 5
```

2.2 Parágrafos

Todos os parágrafos devem ter uma linha entre eles, entre listas e imagens, por exemplo, o texto abaixo:

```
Este é o primeiro parágrafo.
Este é o segundo parágrafo.
```

Será impresso assim:

Este é o primeiro parágrafo. Este é o segundo parágrafo.

2.3 Listas

Listas não ordenadas são criadas utilizando o hífen. O exemplo abaixo é formatado desta forma:

```
- Item 1
- Item 2
- Item 3
```

É exibido desta forma:

- Item 1
- Item 2
- Item 3

Listas ordenadas podem ser criadas utilizando o número na frente do elemento:

```
1. Item 1
2. Item 2
3. Item 3
```

Exibe:

1. Item 1
2. Item 2
3. Item 3

2.4 Tabelas

Sintace:

```
| Head 1 | Head 2 | Head 3 |
| ------ | ------ | ------ |
| Item 1 | Item 2 | Item 3 |
| Item 4 | Item 5 | Item 6 |
| Item 7 | Item 8 | Item 9 |
```

Sendo exibida assim:

Head 1	**Head 2**	**Head 3**
Item 1	Item 2	Item 3
Item 4	Item 5	Item 6
Item 7	Item 8	Item 9

2.5 Links (âncoras)

Links são criados com a seguinte sintaxe:

```
[Texto de exibição](url do caminho)
```

Por exemplo, para acessar o site do Senac ficaria:

```
[Senac](https://www.sp.senac.br/)
```

Seria exibido:

Senac

2.6 Imagens

Sintaxe:

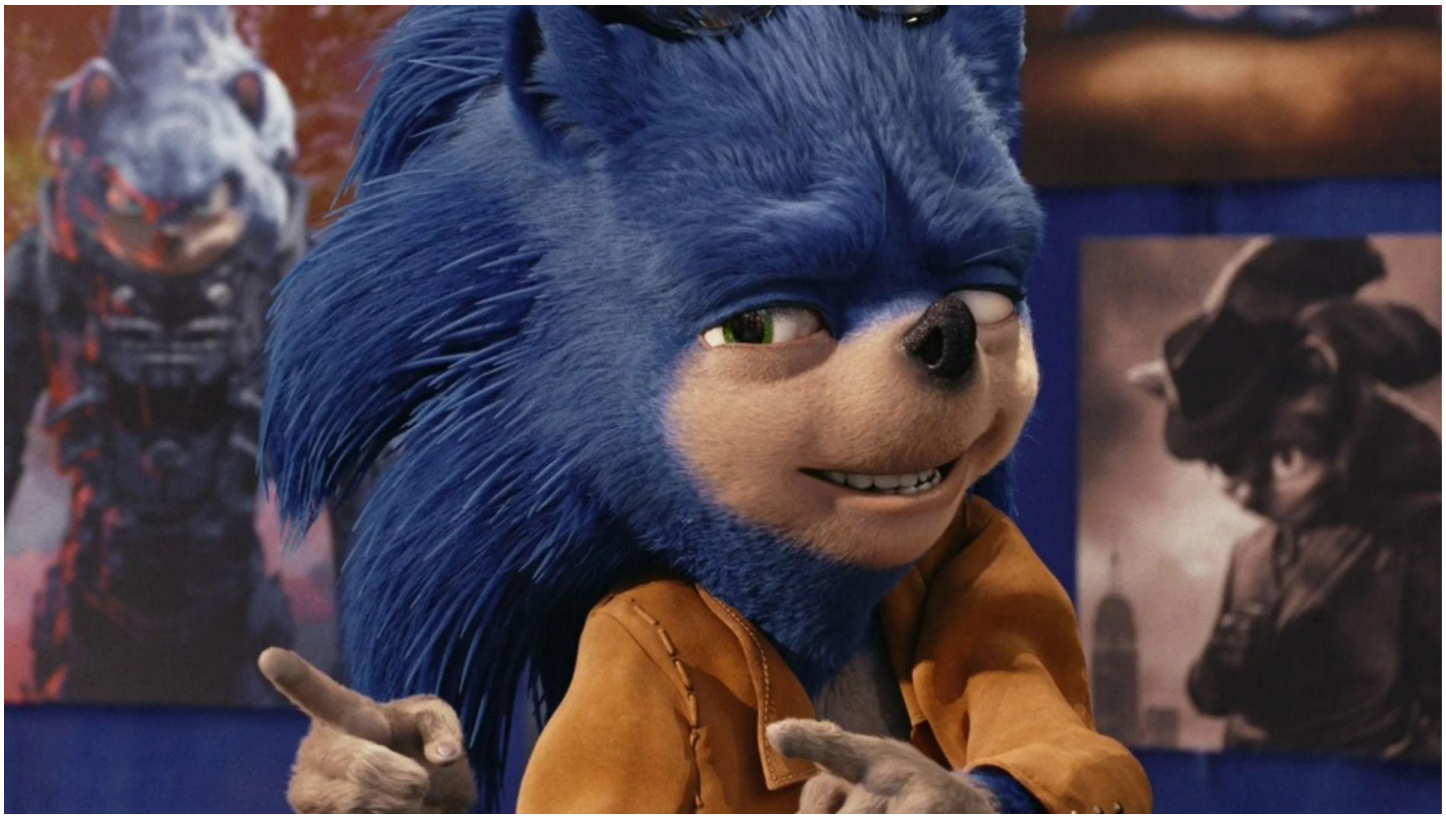
```
![Alt da imagem](url do caminho local ou externa)
```

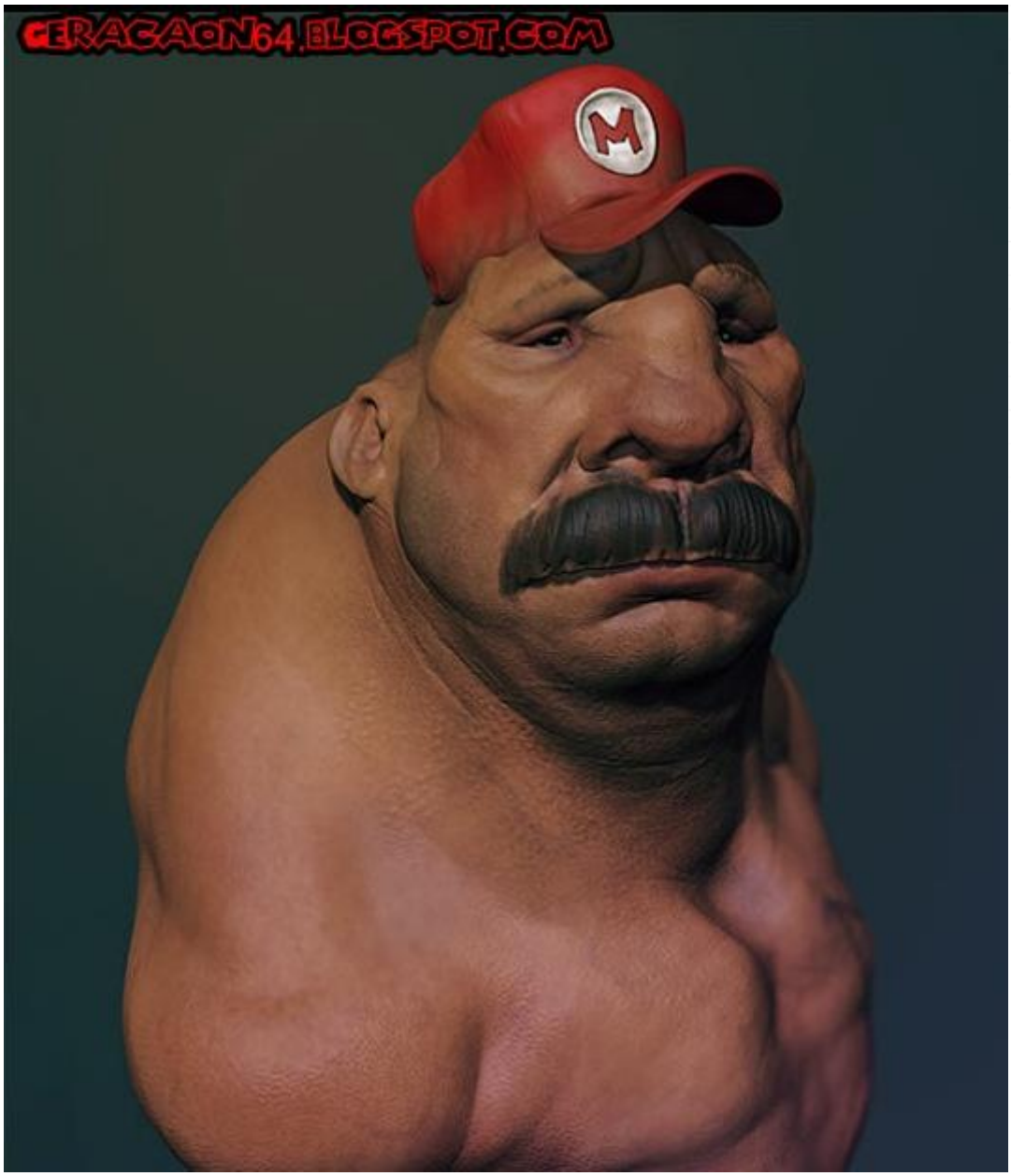
Exemplo:

```
![Exemplo de imagem local](../img/sonic_feio.jpg | 200px)

![Exemplo de imagem externa](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEhbkiAGkj63gvkYUlAJGE-fSnqax1zIIW1QPatBouVPQrK
```

Resultado:





O problema dessa abordagem é que Markdown não suporta ajuste de tamanho de imagem, sendo necessário utilizar html no lugar:

```
<img src="/img/sonic_feio.jpg" width="200">
```

Resultando em:

Exemplo de imagem local com HTML

Para ajustar o tamanho de imagem diretamente pelo Markdown, é necessário instalar alguns plugins. Por exemplo, o Pandoc, ou o Krandown, que permitem ajustar a figura com a sintaxe abaixo (mas é melhor fazer com HTML nesses casos):

```
Pandoc: ![Exemplo de imagem local](../img/sonic_feio.jpg){width=250}
Krandown: ![Exemplo de imagem local](../img/sonic_feio.jpg){: width=250}
```

2.7 Transformando em um projeto HTML

Se desejar, após a construção do documento, é possível transformar o projeto em um projeto HTML.

Navegue, via terminal, até a pasta onde está o arquivo mkdocs.yml. Digite `mkdocs build` e uma pasta chamada `site` será gerada. Nessa pasta todo o conteúdo me foi convertido em HTML + CSS + JavaScript.

2.8 Criando um PDF a partir de um mkdocs

Para criar um PDF são necessários alguns passos mais avançados, mas uma vez configurados, pode-se gerar quantos PDFs quiser para qualquer projeto novo do mkdocs na mesma instalação da máquina.

As dependências necessárias são:

- mintty (através do msys2)
- cairo
- pango
- gdk-pixbuf2
- glib2
- weasyprint
- mkdocs-to-pdf

Instalando o mintty:

- Baixe o msy2 do site https://www.msys2.org/
- Instale a versão correta para o sistema operacional (estará na opção 1: Download the installer)
- Faça a instalação padrão dele.

Instalando o cairo, pango, gdk-pixbuf2 e glib2

Abra o terminal do mintty (MSYS2 MINGW64) e entre com os comandos:

Atualizar o mintty:

```
pacman -Syu
```

Instalar o cairo:

```
pacman -S mingw-w64-x86-cairo
```

Instalar o pango:

```
pacman -S mingw-w64-x86-pango
```

Instalar o gdk-pixbuf2

```
pacman -S mingw-w64-x86-gdk-pixbuf2
```

Instalar o glib2

```
pacman -S mingw-w64-x86-glib2
```

Após a instalação dos pacotes, feche a janela do mintty.

Configurando as variáveis de ambiente

Configure as variáveis de ambiente para que essas bibliotecas fiquem acessíveis ao weasyprint. Os passos para o Windows estão abaixo:

- Abra as variáveis de ambiente e, em variáveis do sistema, crie uma nova variável.
- Para o nome da variável digite: `WEASYPRINT_DLL_DIRECTORIES`
- Para o valor, será incluída a pasta bin da instalação do mintty. Geralmente ela fica em: `C:\msys64\mingw64\bin`.

Feito isso, clique em ok e feche a janela.

Instalando o weasyprint

Abra o terminal no modo de administrador e digite:

```
pip install weasyprint --upgrade
```

Instalando o mkdocs-to-pdf

Ainda no terminal com o modo de administrador:

```
pip install mkdocs-to-pdf
```

No seu projeto mkdocs, abra, no Visual Studio Code, o arquivo `mkdocs.yml` e adicione a dependência do plugin:

```
plugins:
    - to-pdf:
```

Agora, quando rodar o `mkdocs build`, um PDF será criado junto com o site.