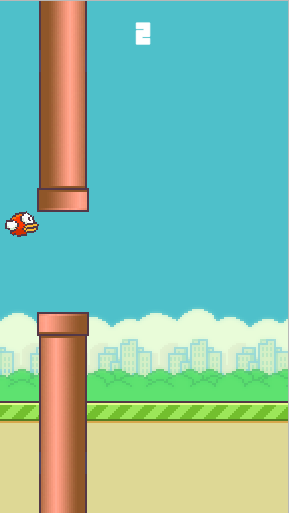
**Tutorial de como compilar um projeto que usa o Qt para WASM**

Nesse tutorial farei a compilação de dois jogos escritos com a biblioteca do qt para WebAssembly:

* **Flappy bird** - Versão do qt: 6.52 <https://github.com/Przemekkkth/FlappyBird_Qt-Cpp>

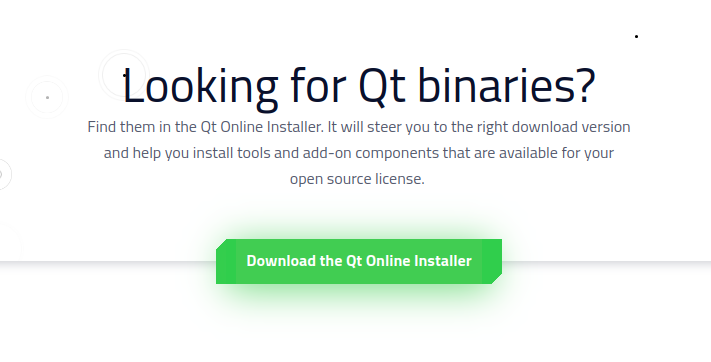


* **PDA MAZE** - Versão do qt: 5.15.2 (Mesma versão do Qt que o MRLeo foi feito)

<https://github.com/EXL/PDA_Maze>



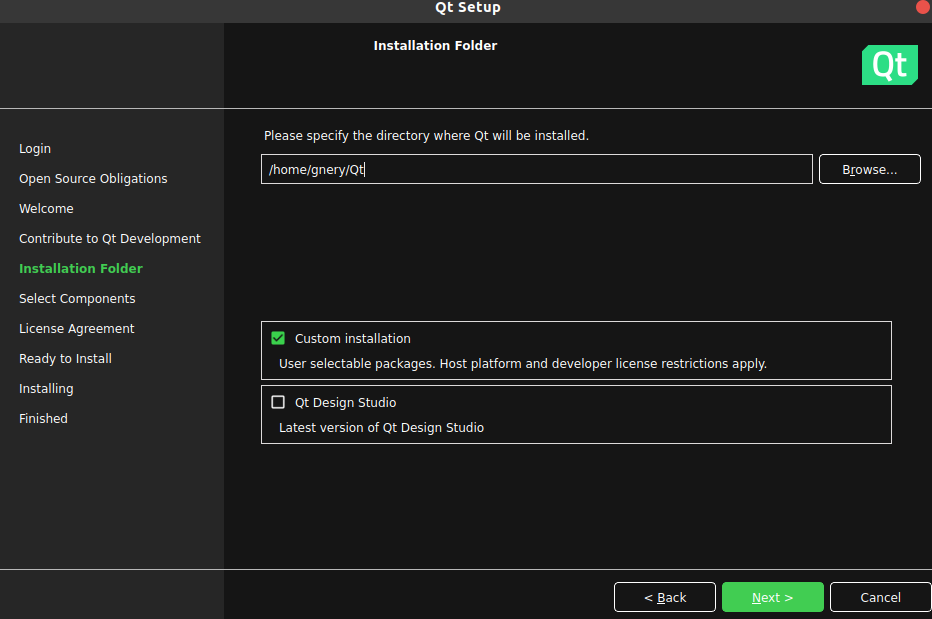
**Obs.:** Clone os repositórios dos jogos para uma pasta onde serão compilados

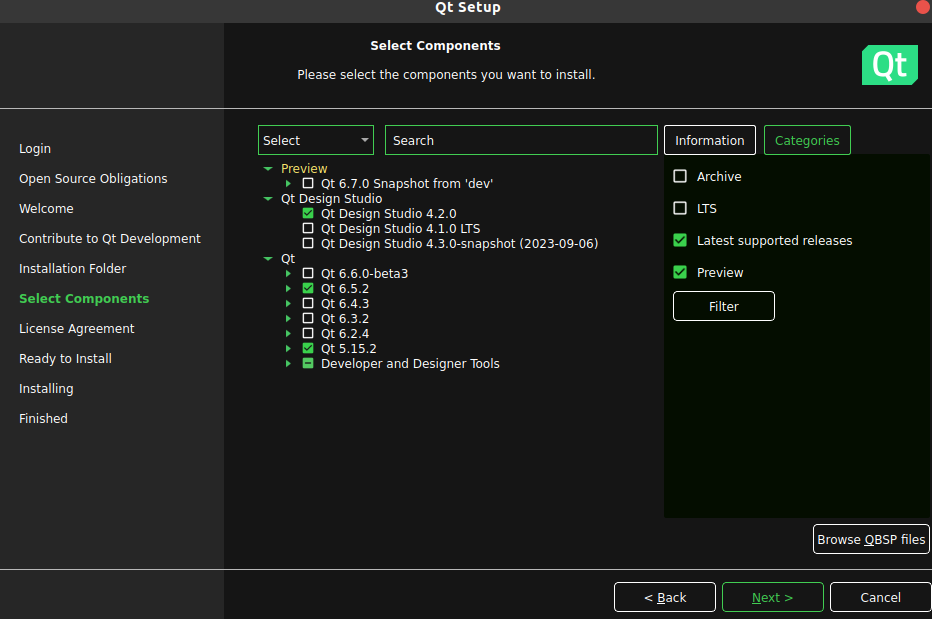
**1- Instalando o Qt** 

* Baixe o instalador do Qt no seguinte site:

**https://www.qt.io/download-open-source**

* Baixe o instalador de acordo com seu sistema operacional e siga as instruções de instalação
* Clique em **instalação customizada**



* Marque as seguintes caixas para instalar o compilador Wasm do Qt, a biblioteca do Qt e o Qt creator (IDE do Qt)

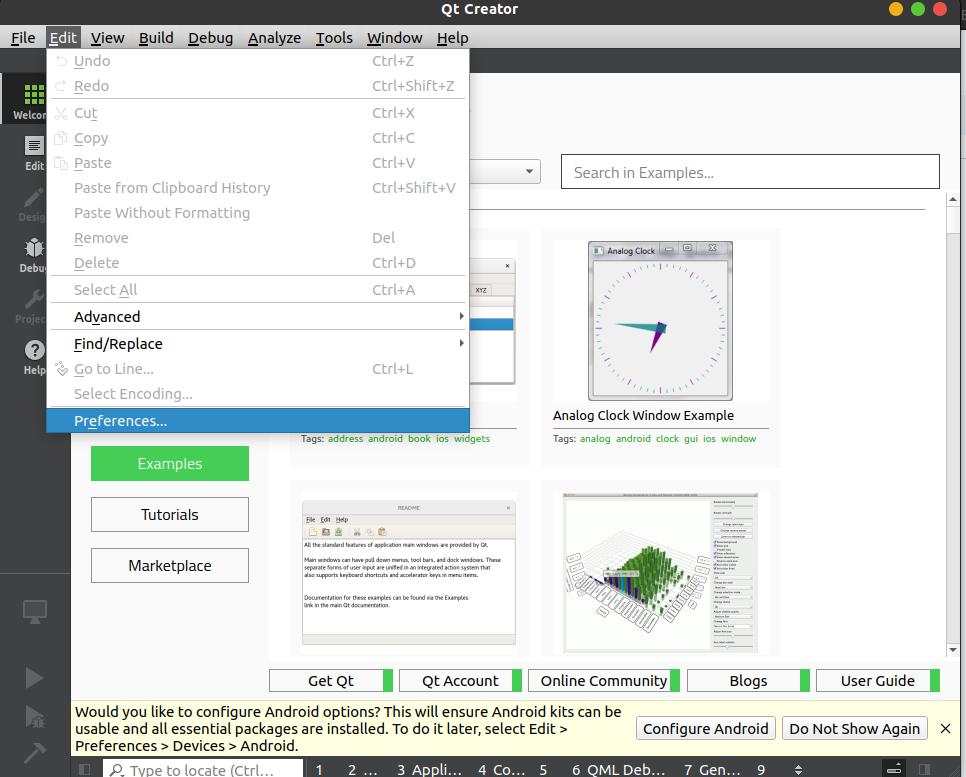
**2- Instalando o emscripten SDK**

* Para instalar o emscripten sdk siga exatamente o tutorial no seguinte link (**Não** use **sudo apt install emscripten**)

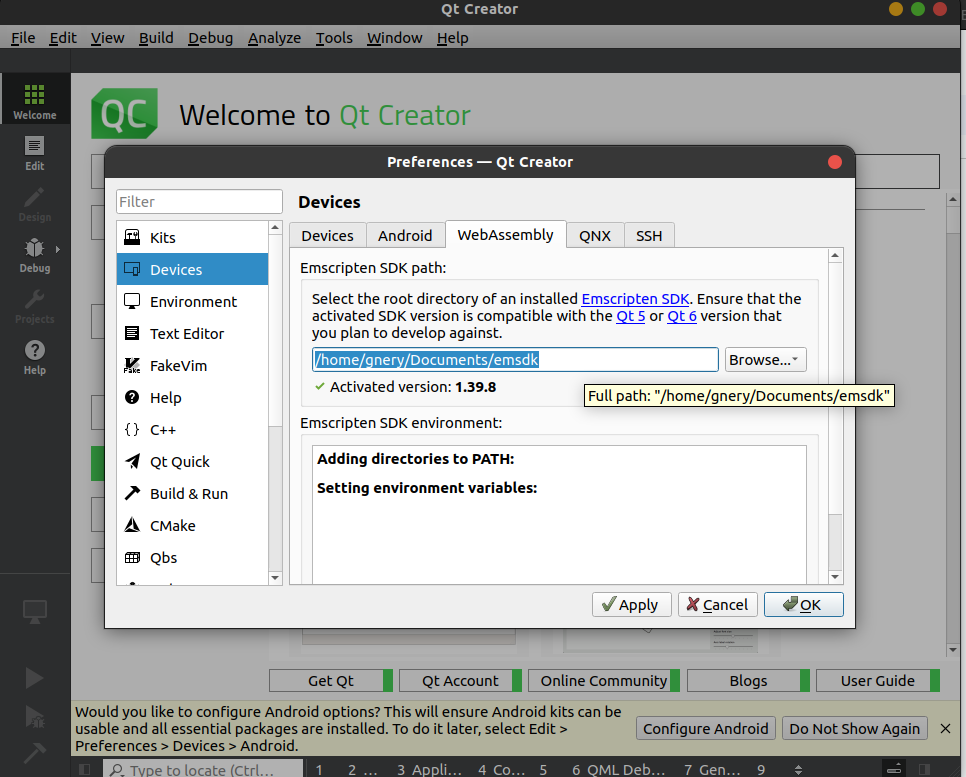
<https://emscripten.org/docs/getting_started/downloads.html>

**3- Configurando o Qt creator para compilar wasm**

* Abra o Qt creator e vá em preferências



* Vá em **devices -> WebAssembly** e coloque o caminho do compilador do emscripten baixado no passo anterior



**4- Compilando o projeto do flappy bird pela IDE do Qt Creator**

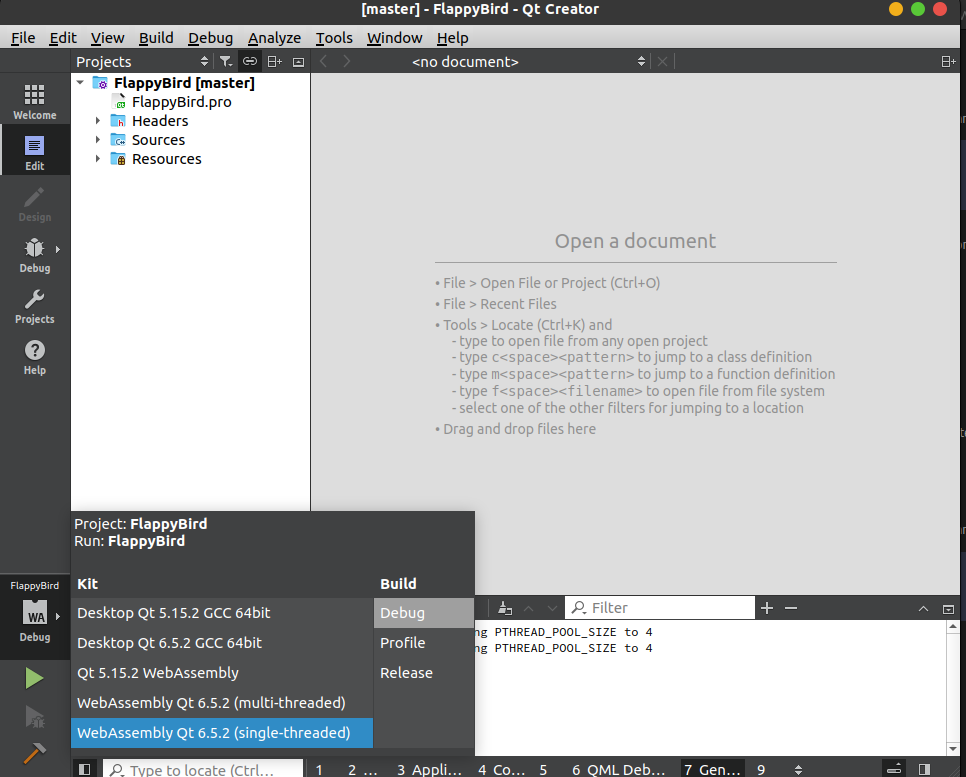
* Instale a versão 3.1.25 do emscripten com:

**./emsdk install 3.1.25**

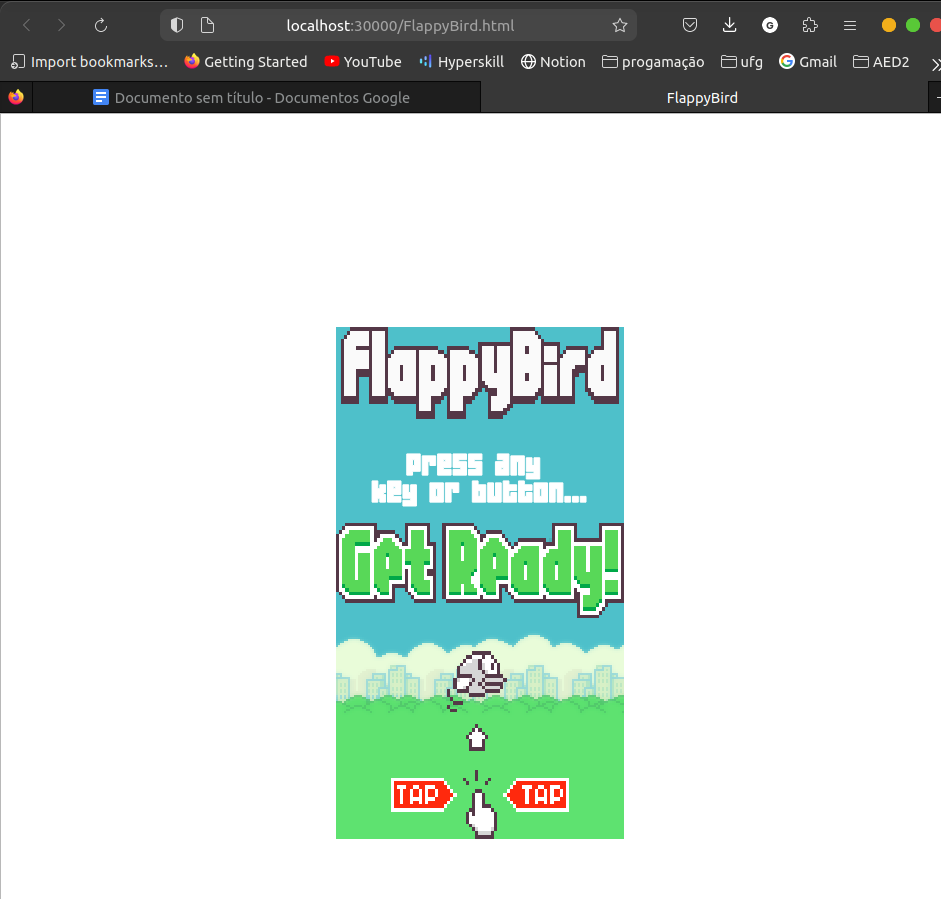
**./emsdk activate 3.1.25**

**Obs.:**Faça isso dentro da pasta do emscripten SDK

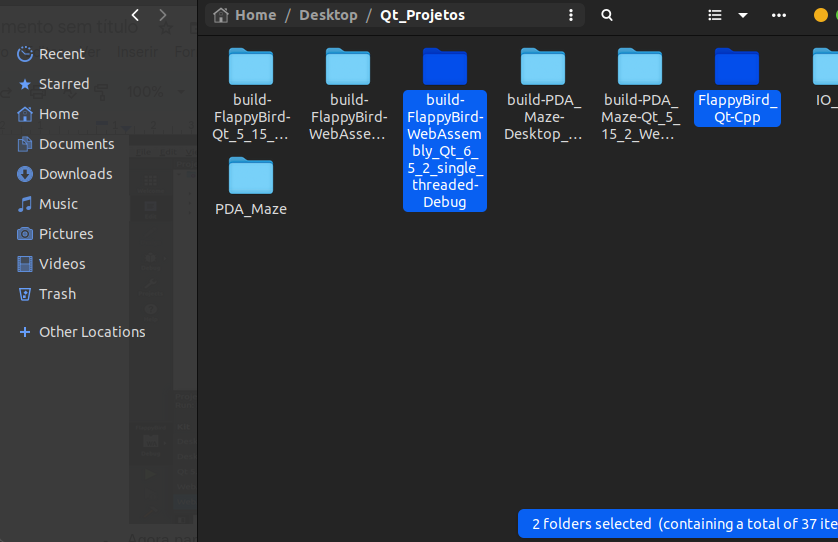
* Vá em Open Project -> FlappyBird\_Qt-Cpp e abra **FlappyBird.pro**
* Vá em debug -> WebAssembly Qt 6.5.2 (single-threaded) e depois clique em build para buildar (Ícone com um martelo) o .wasm, o .html e o .js



* Agora para rodar o projeto basta clicar no botão de play



* Os produtos dessa compilação estão em uma pasta gerada ao lado da pasta fonte do projeto com os arquivos .wasm, .js e .html (build-FlappyBird-WebAssembly\_Qt\_6\_5\_2\_single\_threaded-Debug)



**5- Compilando PDA MAZE pela linha de comando usando o qmake (Sem IDE):**

* Instale a versão 1.39.8 do emscripten com:

**./emsdk install 1.39.8**

**./emsdk activate 1.39.8**

**Obs.:**Faça isso dentro da pasta do emscripten SDK

* Encontre a pasta do “qmake-webassembly” associado a versão 5.15.2 do qt, a minha está em:

**/home/gnery/Qt/5.15.2/wasm\_32/bin**

* Volte a pasta do projeto do jogo e execute onde o arquivo .pro está :

**/home/gnery/Qt/5.15.2/wasm\_32/bin/qmake**

gerando o arquivo Makefile

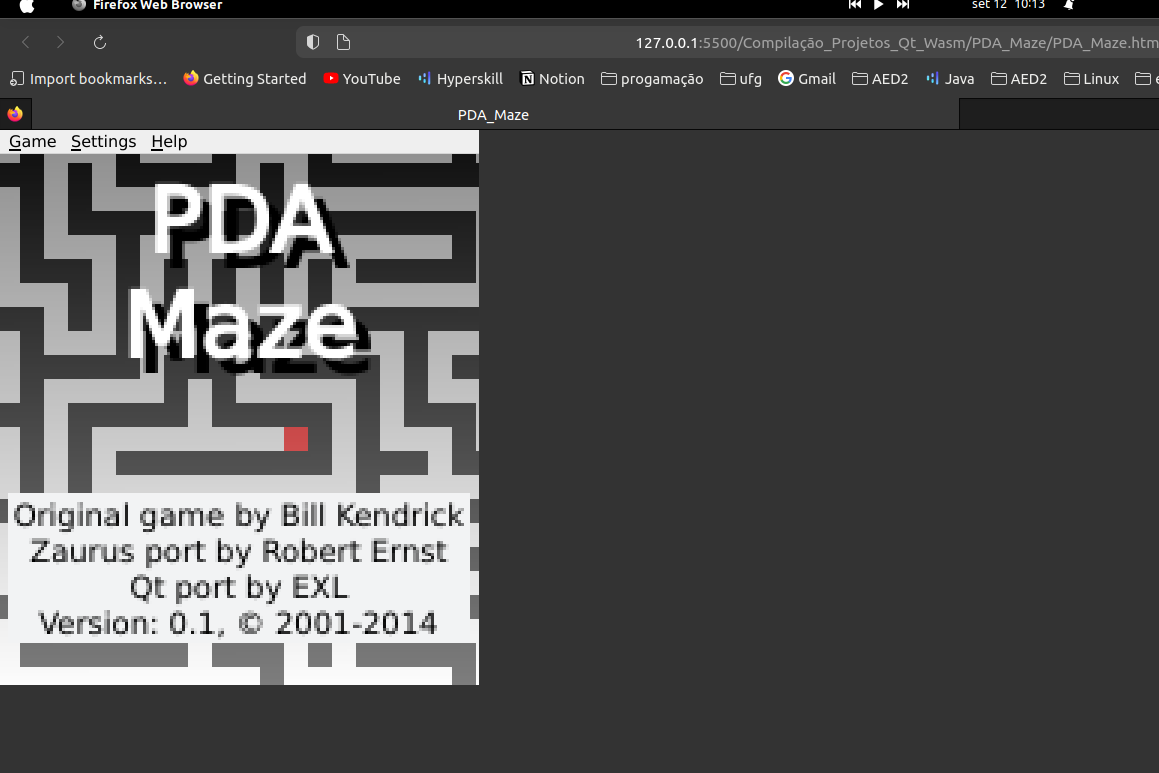
* Por fim, execute na mesma pasta:

**make**

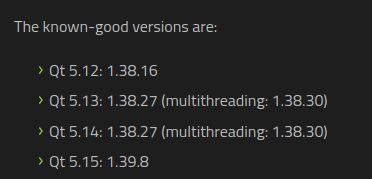
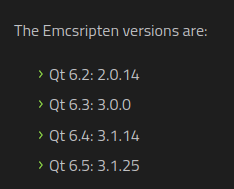
* (Opcional) execute **make clean** para retirar os arquivos .o

**6- Executando o jogo**

* Para executar o jogo, basta rodar o .html gerado em um servidor local (extensão do vc code já faz isso)

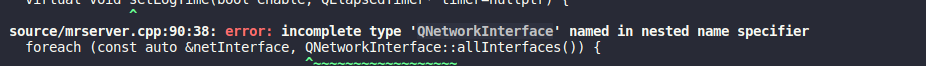


**Observações**

* Esses são, atualmente, os únicos jeitos de compilar um projeto que usa a biblioteca do Qt, para wasm
* Aparentemente o MRLeo foi completamente desenvolvido no Qt creator, uma vez que ele possui um arquivo .pro, o qual é compilado apenas pelo qmake do Qt
* Cada versão do Qt precisa de uma versão específica do compilador do emscripten:

**Compilando o MRLeo com qmake para wasm**

Ao compilar o código do MRLeo baseado nos passos realizados acima, fiz todas as recomendações que o compilador foi me sugerindo, como incluir certas bibliotecas ao cpath e retirar certos parâmetros de compilação. Porém ocorreu um grave problema: A classe **QNetworkInterface** (Abstrai funções e parâmetros de rede), não foi portada para o nenhuma versão do Qt WebAssembly, logo para compilar o MRLeo para wasm seria necessário algumas alternativas:



1. Procurar uma biblioteca de rede externa para substituir as funcionalidades de rede que a biblioteca do Qt usa. Essa biblioteca deve ser portável para wasm
2. Podemos usar um script externo em js que faz substitui essas funcionalidades do Qt
3. Podemos fazer um cliente cru, sem nenhuma função de rede implementada

**Outras observações:**

O projeto do MRLeo usa multi-thread, porém a versão 5 do qt, versão usada pelo MRLeo, não suporta esse recurso, logo o resultado final será lento

**Possível solução:**

Portar o código do Qt 5 para Qt 6 no projeto do MRLeo