***Aula Phaser***

***O que é ?***

O Phaser é um framework, ele nos auxiliará a criar nossos jogos em JavaScript. É um pouco difícil diferenciar um framework de uma biblioteca, eu diria que para desenvolver usando um framework, você tem que **pensar** da maneira “imposta” por ele.

Por exemplo, passamos bastante tempo estudando JavaScript, porém o grau de abstração do Phaser é tão grande, que agora devemos seguir as regras do Phaser ( esse que é desenvolvido em JavaScript ) para alcançar nossos objetivos.

***Instalação***

Iremos fazer o download do Phaser em seu site oficial: phaser.io, usaremos a versão: min.js. Essa versão vem com tudo pronto para ser usado. Em seguida, no html declaramos ele:

<script sc= “./min.js”></script>

Para o nosso framework funcionar, é preciso roda-lo em um servidor, por motivos de segurança o browser não aceita diretórios do tipo: “file://” Mas isso não será um problema, pois a nossa extensão *live server* já realiza isso para a gente.

**OBS:** Estaremos trabalhando com a versão 3.22.0 do Phaser

***Primeiros passos***

Nesse documento abordarei o básico necessário para começarmos a programar no Phaser, além do mais, isso poderia facilmente virar uma biblioteca de comandos. Não há melhor forma de aprender do que a prática !

Primeiramente criaremos as “scenes”, elas serão responsáveis por executar as ações do jogo, podemos ter várias scenes ao mesmo tempo, entretanto no nosso caso usaremos algumas para os níveis do jogo.

let gameScene = new Phaser.Scene(“Game”);

Em seguida, devemos criar o objeto Phaser.Game(parâmetro) e passar as configuração como parâmetro:

let config = {  
 type: Phaser.auto, //Define a tecnologia de renderização, WebGL ou Canvas

width: 800, //Largura

height: 600, //Altura

scene: gameScene;

}

let game = new Phaser.Game(config);

Depois definimos os comportamentos em seus devidos lugares, como será visto na aula.

***Como funciona o GameLoop do Phaser***

A nossa cena precisa ter 3 funções/métodos:

Init: init => É chamado uma vez quando o jogo carregar

preload: preload, => É chamado uma vez, é responsavel por carregar imagens, sons... etc

create: create, => É chamado uma vez , reponsável por criar os elementos na tela

update: update => Responsável por atualizar as informações