Adicionando funcionalidades do Jogo

Se cada evento for descrito com muitos detalhes como antes, teremos um tutorial muito longo. Vamos tornar a descrição um pouco mais breve para os próximos eventos. Lembre-se de que as etapas para adicionar uma condição ou ação são:

1. Clique duas vezes para inserir um novo evento ou clique no link Adicionar ação para adicionar uma ação.

2. Clique duas vezes no objeto em que a condição/ação está.

3. Clique duas vezes na condição/ação desejada.

4. Insira os parâmetros, se algum for necessário.

De agora em diante, os eventos serão descritos como o objeto, seguidos pela condição/ação, seguidos por quaisquer parâmetros. Por exemplo, o evento que acabamos de inserir poderia ser escrito:

Add condition *System* -> *Every* tick

Add action *Player* -> Definir ângulo para a posição -> X: *Mouse.X*, Y: *Mouse.Y*

Fazer o player atirar

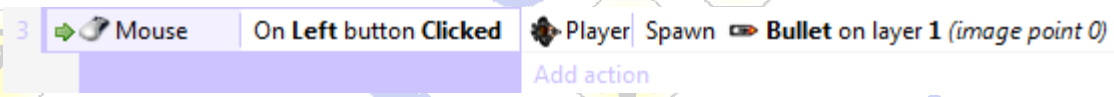
Quando o jogador clica, o *player* deve atirar uma bala. Isso pode ser feito com a ação *Spawn an object* no Player, que cria uma nova instância de um objeto na mesma posição e ângulo. O *Bullet movement* já havíamos adicionado antes, então o faremos voar para frente.

Faça o seguinte evento:

Condição: *Mouse* -> *On click* -> Left clicked (the default)
Ação: *Player* -> *Spawn another object* -> Para *Object*, escolha o objeto *Bullet*.

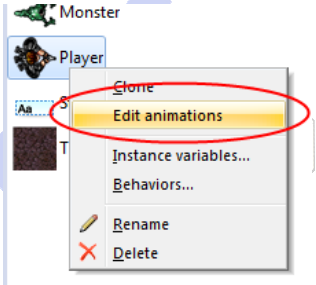
Para *Layer*, colocar **1** (o layer "Main" é o layer 1- lembre-se que Construct2 conta a partir de zero). Deixe *Image point* como 0.

Seu evento agora deve estar assim:

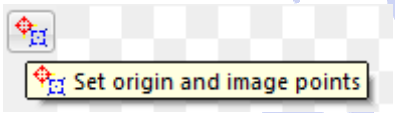


Se você executar o jogo, você notará as balas dispararem do meio do jogador, ao invés do final da arma. Vamos consertar isso colocando um **image point** no final da arma. (Um ponto de imagem é apenas uma posição em uma imagem da qual você pode gerar objetos.)

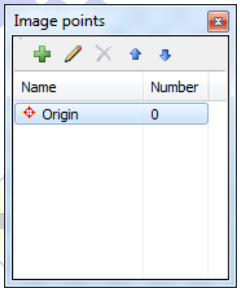
Clique com o botão direito no Player no projeto ou na barra de objetos e selecione **Edit animations**.



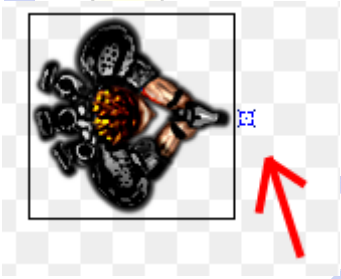
O editor de imagens para o jogador reaparece. Clique na ferramenta de origem e pontos de imagem:



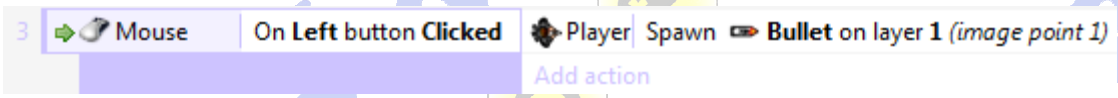
... e a caixa de diálogo de pontos de imagem é aberta



Observe que a origem do objeto aparece como um ponto vermelho. Essa é a "hotspot" ou "pivot point" do objeto. Se você girar o objeto, ele gira em torno da origem. Queremos adicionar outro ponto de imagem para representar a arma, então clique no verde *add*button. Um ponto azul aparece - esse é o nosso novo ponto de imagem. Clique com o botão esquerdo no final da arma do jogador para colocar o ponto da imagem lá:



Feche o editor de imagens. Clique duas vezes no *Spawn an object* action que adicionamos anteriormente e troque *Image point* para **1**. (A origem é sempre o primeiro ponto da imagem, e lembre-se que o Construct 2 conta a partir do zero.) O evento deve agora parecer abaixo - note que ele diz Image *point 1* agora:



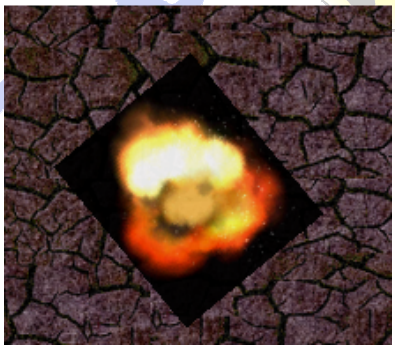
Execute o jogo. As balas agora disparam do final da sua arma! As balas ainda não fazem nada. Espero que, no entanto, você comece a perceber que, assim que se familiarizar com o sistema de eventos, poderá colocar a funcionalidade em funcionamento muito rapidamente.

Vamos fazer as balas matar monstros. Adicione o seguinte evento:

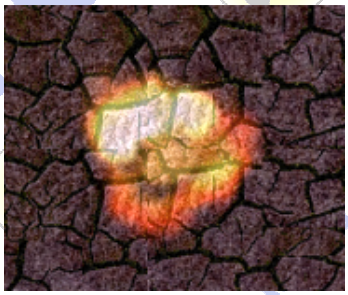
Condition: *Bullet -> On collision with another object* -> pick *Monster*.
Action: *Monster -> Destroy*
Action: *Bullet -> Spawn another object -> Explosion*, layer **1**
Action: *Bullet -> Destroy*

The explosion effect

Rode o jogo e tente atirar em um monstro. Oops, a explosão tem aquela grande borda preta!



Você pode ter previsto que ficaria assim desde o começo, e imaginou se o nosso jogo realmente terminaria assim! Não se preocupe, não vai. Clique em **Explosion object** na barra de objetos no canto inferior direito ou na barra de projeto (que foi tabulada com a barra de camadas). Suas propriedades aparecem na barra de propriedades à esquerda. Na parte inferior, defina as propriedades de **Blend mode** para **Additive**. Agora rode o jogo novamente.

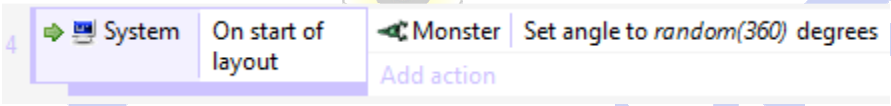


Por que isso funciona? Sem entrar nas porcas e parafusos, as imagens comuns são coladas no topo da tela. Com o efeito aditivo, cada pixel é adicionado (como em, soma) com o pixel de fundo atrás dele. O preto é um valor de pixel zero, então nada é adicionado - você não vê o fundo preto. Cores mais brilhantes adicionam mais, então aparecem mais fortemente. É ótimo para explosões e efeitos de iluminação.

Making monsters a little smarter

Neste momento os monstros apenas se afastam do layout para a direita. Vamos torná-los um pouco mais interessantes. Primeiro de tudo, vamos iniciá-los em um ângulo aleatório.

Condition: *System -> On start of Layout*
Action: *Monster -> Set angle* -> random(360)



Eles ainda irão embora para sempre quando deixarem o layout, para nunca mais serem vistos novamente. Vamos mantê-los dentro. O que faremos é apontá-los de volta para o jogador quando eles saírem do layout. Isso faz duas coisas: eles sempre ficam dentro do layout, e se o jogador parar, os monstros vêm direto para ele!

Condition: *Monster -> Is outside layout*
Action: *Monster -> Set angle toward position* -> For X, **Player.X** - for Y, **Player.Y**.

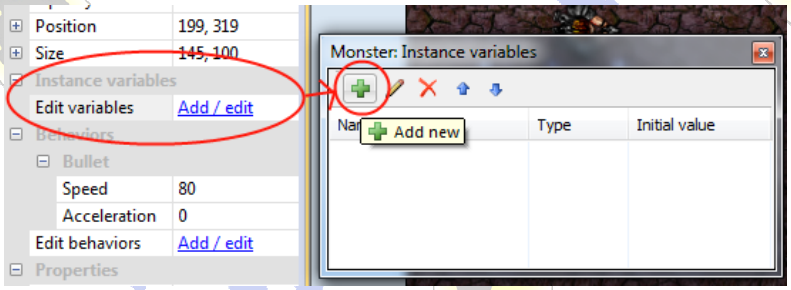
Execute o jogo. Se você andar por ai por um tempo, perceberá que os monstros também ficarão ao redor do layout e estarão indo em todos os sentidos. É um trabalho de inteligência artificial (AI) difícil, mas irão fazer!

Agora, suponha que tenhamos que atirar em um monstro cinco vezes antes que ele morra, ao invés da morte instantânea como ocorre no momento. Como fazemos isso? Se nós apenas criarmos um contador de "Vida", então uma vez que atingimos um monstro cinco vezes, todos os monstros vão morrer. Em vez disso, precisamos que cada monstro se lembre de sua própria saúde. Nós podemos fazer isso com **instance variables**.

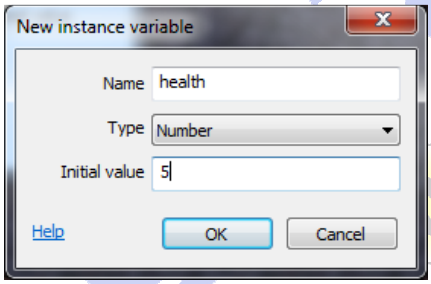
Instance variables

Variáveis de instância permitem que cada monstro armazene seu próprio valor de vida. Uma variável é simplesmente um valor que pode mudar (ou variar), e eles são armazenados separadamente para cada instância, daí o nome *instance variable*.

Vamos adicionar um *health* variável de instância para o nosso monstro. Clique no monstro na barra de projeto ou na barra de objetos. Alternativamente, você pode voltar para o layout e selecionar um objeto monstro. Isso mostrará as propriedades do monstro na barra de propriedades. Clique **Add/edit** de **Edit variables**.

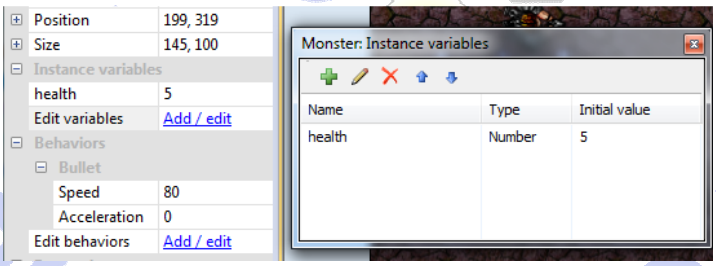


A caixa de diálogo Instance Variables é exibida. Parece semelhante a caixa de diálogo Behaviors que vimos anteriormente, mas permite adicionar e alterar variáveis de instância para o objeto. Clique no verde **Add** botão para adicionar um novo.



Na caixa de diálogo que aparece, digite **health** para o nome, deixe *Type* como **Number** e para *Initial value* coloque **5**. Isso inicia cada monstro com 5 pontos de vida. Quando eles forem atingidos, vamos subtrair 1 da saúde e, quando a saúde for zero, destruiremos o objeto.

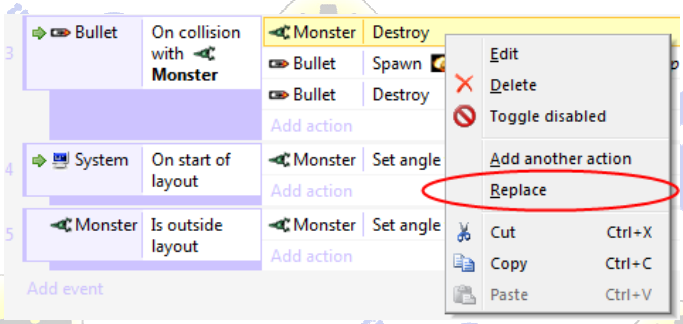
Quando terminar, clique em OK. Observe que a variável agora aparece na caixa de diálogo instance variables e também nas propriedades do monstro. (Você pode alterar rapidamente os valores iniciais na barra de propriedades, mas para adicionar ou remover variáveis, será necessário clicar no botão Add/*Edit*.)



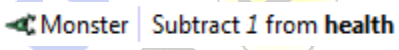
Changing the events

Volte para event sheet. No momento, estamos destruindo monstros assim que a bala os atinge. Vamos mudar isso para subtrair 1 da sua saúde.

Encontre o evento que diz: *Bullet - on collision with Monster*. Observe que temos uma ação "destroy monster". Vamos substituir isso por " subtract 1 from health ". Clique com o botão direito na ação "destroy monster" e clique em **Replace**.

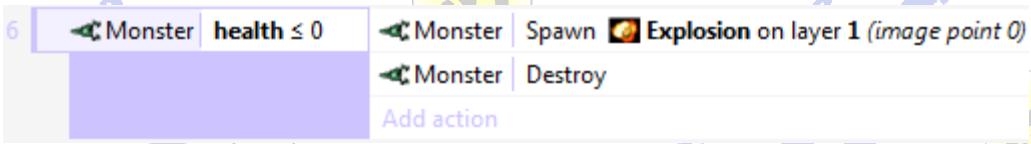


A mesma caixa de diálogo aparece como se estivéssemos inserindo uma nova ação, mas dessa vez substituirá a ação em que clicarmos. Escolher *Monster -> Subtract from* (na categoria *Instance variables*) -> Instance variable "health", e insira **1** para *Value*. Clique **Done**. Action deve estar assim:



Agora, quando atirarmos nos monstros, eles perderão 1 ponto de vida e a bala explodirá, mas não fizemos um evento para matar monstros quando a saúde deles chega a zero.Então, adicione outro evento:

Condition: *Monster -> Compare instance variable* -> Health, *Less or equal*, 0
Action: *Monster -> Spawn another object* -> Explosion, layer **1**
Action: *Monster -> Destroy*

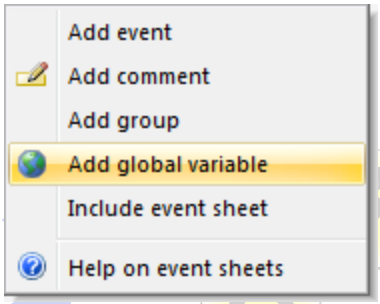


Por quê "less or equal 0" ao invés de "equals 0"? Suponha que adicionamos outra arma mais poderosa que subtraiu **2** da vida. Como você atirou em um monstro, sua vida seria **5**, **3**, **1**, **-1**, **-3**... observe que em nenhum momento sua vida ficou *equal to zero*, então, nunca morreria! Portanto, é uma boa prática usar "less or equal" para testar se a vida de alguma coisa acabou.

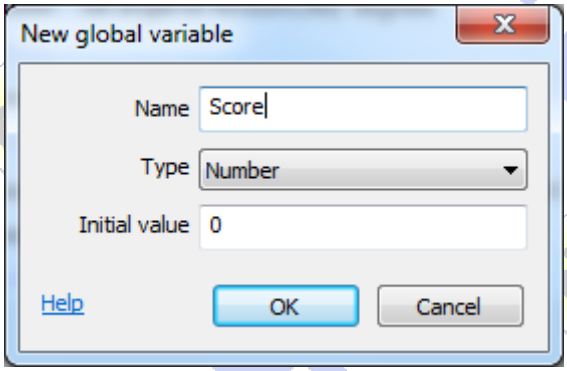
Execute o jogo. Você agora tem que acertar monstros cinco vezes para matá-los!

Keeping score

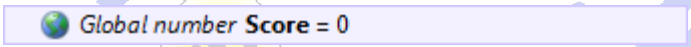
Vamos ter uma pontuação para que o jogador saiba o quão bem ele está. Vamos precisar de outra variável para isso. Podemos usar uma **global variable**. Como uma variável de instância, uma variável global (ou apenas "global") pode armazenar texto ou um número. Cada variável pode armazenar um único número ou uma única parte do texto. Variáveis globais também estão disponíveis para o jogo inteiro em todos os layouts - O que é conveniente se adicionarmos outros níveis. Clique com o botão direito em algum espaço na parte inferior da event sheet e selecione *Add global variable*.



Coloque **Score** como nome. Os outros padrões de campo estão OK, isso fará com que seja um número a partir de 0.

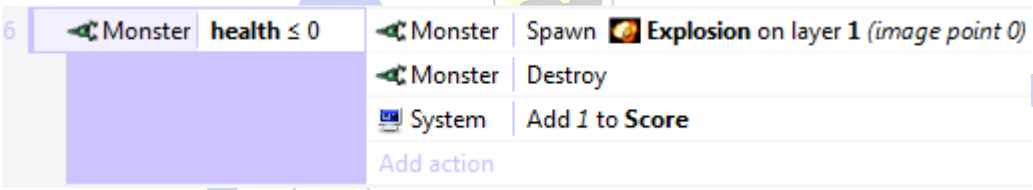


Agora, a variável global aparece como uma linha na event sheet. Está nesta event sheet, mas pode ser acessada de qualquer event sheet em qualquer layout.



Note: também existem variáveis locais que só podem ser acessadas por um "scope" de eventos, mas não precisamos nos preocupar com isso agora.

Vamos dar ao jogador um ponto para matar um monstro. Na nossa "Monster: health less or equal 0" event (quando um monstro morre), clique **Add action**, e selecione *System* -> *Add to* (em variáveis globais e locais) -> **Score**, valor **1**. Agora o evento deve ficar assim:



Agora o jogador tem uma pontuação, que aumenta em 1 para cada monstro que ele mata - mas eles não podem ver sua pontuação! Vamos mostrar a pontuação com um objeto de texto.

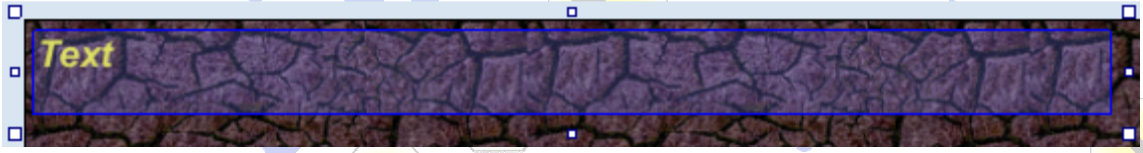
Creating a heads-up display (HUD)

Um heads-up display (também conhecido como HUD) é a interface que mostra a vida do jogador, a pontuação e outras informações do jogo. Vamos fazer um HUD realmente simples a partir de um objeto de texto.

O HUD sempre fica no mesmo lugar na tela. Se tivermos alguns objetos de interface, não queremos que eles se afastem enquanto o jogador caminha - eles devem permanecer na tela. Por padrão, as camadas rolam. Para mantê-los na tela, podemos usar a configuração de camada **Parallax**. O Parallax permite que diferentes camadas rolem em taxas diferentes para uma espécie de efeito semi-3D. Se definirmos a paralax como zero, a camada não rolará de maneira alguma - ideal para um HUD.

Volte para a barra de camadas que usamos anteriormente. Adicione uma nova camada chamada *HUD*. Certifique-se de que esteja no topo e selecionada (lembre-se que isso torna a camada ativa). A barra Propriedades deve agora exibir suas propriedades. Coloque a propriedade **Parallax** para **0, 0** (isso é zero em ambos os eixos X e Y).

De um duplo clique em um espaço para inserir outro objeto. Desta vez, escolha o objeto **Text**. Coloque-o no canto superior esquerdo do layout. Vai ser difícil ver se for preto, por isso, na barra de propriedades, use negrito, itálico, amarelo e escolha um tamanho de fonte um pouco maior. Redimensione-o o suficiente para caber uma quantidade razoável de texto. Deve ser algo como isto:



Volte para event sheet. Vamos manter o texto atualizado com a pontuação do jogador. No evento **Every tick** que adicionamos anteriormente, adicione a ação *Text -> Set text.*

Use o operador **&** , podemos converter um número em texto e juntá-lo a outra cadeia de texto. Então, para o texto, digite:

"Score: " & Score

A primeira parte (*"Score: "*) significa que o texto sempre começará com a frase Score :. A segunda parte (*Score*) é o valor real da variável global Pontuação. O **&** junta-os em um pedaço de texto.

Execute o jogo e atire em alguns monstros. Sua pontuação é exibida e fica no mesmo lugar na tela!

Finishing touches

Estamos quase terminando. Vamos adicionar alguns toques finais.

Primeiramente, vamos ter alguns monstros nascendo regularmente, pois caso você atire em todos os monstros, não haverá mais nada a fazer. Então, vamos criar um novo monstro a cada 3 segundos.

Adicione um novo evento:

Condition: *System -> Every X seconds* -> **3**
Action: *System -> Create object* -> **Monster**, layer **1**, **1400** (for X), **random(1024)**(for Y)

1400 é uma coordenada X ao lado da borda direita do layout, e *random(1024)* é um coordenada Y aleatória, que define a altura no layout.

Finalmente, vamos ter monstros eliminando o jogador.

Condition: *Monster -> On collision with another object ->* **Player**
Action: *Player -> Destroy*

Conclusion

Parabéns, você fez seu primeiro jogo HTML5 no Construct 2! Se você tem um servidor e quer mostrar seu trabalho, clique em **Export** no menu Arquivo. O Construct pode salvar todos os arquivos do projeto em uma pasta no seu computador, que você pode carregar ou integrar em uma página da web.

Você aprendeu algumas noções básicas importantes sobre o Construct2: inserir objetos, usar camadas, comportamentos, eventos e muito mais. Espero que isso o deixe bem preparado para aprender mais sobre o Construct 2! Tente explorar seus recursos e veja o que ele pode fazer por você.