

Jogos 2D com Stencyl

Crie jogos completos sem códigos de programação



Sumário

- ISBN
- Apresentação
- 1. Primeiros passos
- 2. Utilizando behaviors no jogo
- 3. Trabalhando com colisões
- 4. Criando behaviors
- 5. Cronometrando o tempo no jogo
- 6. Animações e efeitos visuais
- 7. Vidas e Game Over
- 8. Criando mais níveis
- 9. Finalizando o jogo
- 10. Testando e publicando o jogo

ISBN

Impresso e PDF: 978-85-7254-045-2

EPUB: 978-85-7254-046-9

MOBI: 978-85-7254-047-6

Caso você deseje submeter alguma errata ou sugestão, acesse
<http://erratas.casadocodigo.com.br>.

Apresentação

Stencyl é um software para o desenvolvimento de games 2D utilizado tanto por profissionais como amadores. O método utilizado para a criação dos jogos é o *drag-and-drop*, arrastar e soltar, ou seja, as linhas de códigos de programação foram substituídas por pequenos blocos coloridos para a montagem das instruções, facilitando sua utilização. A interface amigável e a diferenciação das categorias de instruções por cores auxiliam na compreensão e no desenvolvimento das tarefas.

Personagens, cenários e níveis do jogo são importados e utilizados no jogo dessa forma, sem a necessidade sequer de utilização de uma linha de código de programação. Sendo assim, o Stencyl proporciona o desenvolvimento de um jogo completo na estética 2D, permitindo o acesso a um público maior que quer desenvolver seus próprios projetos. Diversos tipos de jogos no mercado possuem essa estética, tanto apenas para entretenimento como os chamados *Serious Games*, cuja proposta vai além do entretenimento, ou seja, jogos que se propõem a mudar comportamentos, seja contribuir para motivar o jogador a realizar uma certa atividade, como praticar esportes, ou motivá-lo para uma causa mais humanitária, como doar sangue ou conservar o planeta.

O livro está dividido basicamente em 5 etapas:

1. Na primeira, você conhecerá a interface do software e como funciona a utilização dos blocos de instrução;
2. Em seguida, será apresentado como incluir uma cena, ou seja, como iniciar a realização do primeiro nível do jogo, a inclusão de personagens e objetos para compor o cenário;
3. Após a elaboração da primeira cena, são apresentadas as primeiras instruções: movimentação do personagem, suas colisões com outros elementos e contagem regressiva. Essa etapa é bastante importante pois você já percebe a jogabilidade

funcionando e vê que é possível desenvolver um jogo sem a necessidade de programação;

4. Nessa etapa do livro você já compreendeu como funciona o método de desenvolvimento do Stencyl. Então, algumas instruções mais elaboradas são apresentadas, como criação de outros níveis, chances de vida para o personagem e finalização e reinicialização do jogo;
5. A etapa final do livro é o desenvolvimento das telas iniciais e finais, a inclusão dos efeitos de som e trilha sonora e informações como divulgar o jogo nas plataformas de distribuição.

Todo o processo de desenvolvimento do livro foi realizado de maneira descomplicada com uma didática que facilite a compreensão dos recursos utilizados no software. O Stencyl abre possibilidades para que você consiga desenvolver e finalizar seu jogo, capacitando-o para, no futuro, utilizar-se dessa ferramenta na produção de seus projetos para, também, atuar no mercado de trabalho. E sem a necessidade de linguagens de programação complexas, bastando como base principal o conhecimento teórico e conceitual que é fundamental para que se realize um bom projeto de jogo.

As principais características do Stencyl são:

1. Rápido desenvolvimento do protótipo do jogo

Com poucos e simples recursos, você conseguirá desenvolver a jogabilidade inicial do seu jogo, como criação de cenas (níveis) e movimentação do personagem.

2. Jogos desenvolvidos sem a necessidade de linhas de código de programação

O Stencyl é o software ideal para aqueles que não possuem conhecimento de códigos de programação. As linhas de código são

substituídas por blocos de instrução coloridos.

3. Interface intuitiva

Desenvolver jogos com Stencyl é muito simples. Sua convidativa interface intuitiva facilita ao usuário encontrar os recursos necessários para a criação do jogo.

O Stencyl pode ser instalado nos sistemas operacionais Microsoft Windows, Mac OS e Linux. Os dispositivos e sistemas operacionais onde os jogos podem rodar são:

- iOS (iPod, iPhone e iPad)
- Android (Smartphones e tablets)
- Navegadores com suporte a HTML5
- Desktops Microsoft Windows XP em diante
- Desktops Linux
- Desktops Mac OS X em diante

Para a prática do software, você pode utilizar a versão gratuita. No entanto, existem algumas limitações:

- Os jogos só podem ser publicados em Flash e HTML5.
- Telas personalizadas de carregamento do jogo não podem ser implementadas.
- Jogos podem ser criados e desenvolvidos para App Store, mas não publicados.
- Inserção de marca-d'água em algumas plataformas de publicação.

Para a atualização das informações acima, acesse a página do Stencyl em: <http://www.stencyl.com>.

Como instalar o Stencyl

1. Acesse o site do Stencyl;

2. Procure a opção `Download` ;
3. Após o clique, selecione o sistema operacional e clique novamente em `Download` ;
4. Acesse a pasta onde o arquivo de instalação foi baixado e dê um clique duplo sobre ele. Siga as orientações de instalação até sua conclusão.

CAPÍTULO 1

Primeiros passos

1.1 Conhecendo a interface do Stencyl

Vamos iniciar nossa jornada pelo software conhecendo primeiramente suas duas interfaces básicas: a tela de abertura do programa e, depois, a tela de abertura do jogo. Quando abrirmos o Stencyl, aparecerá a interface a seguir com as seguintes opções:

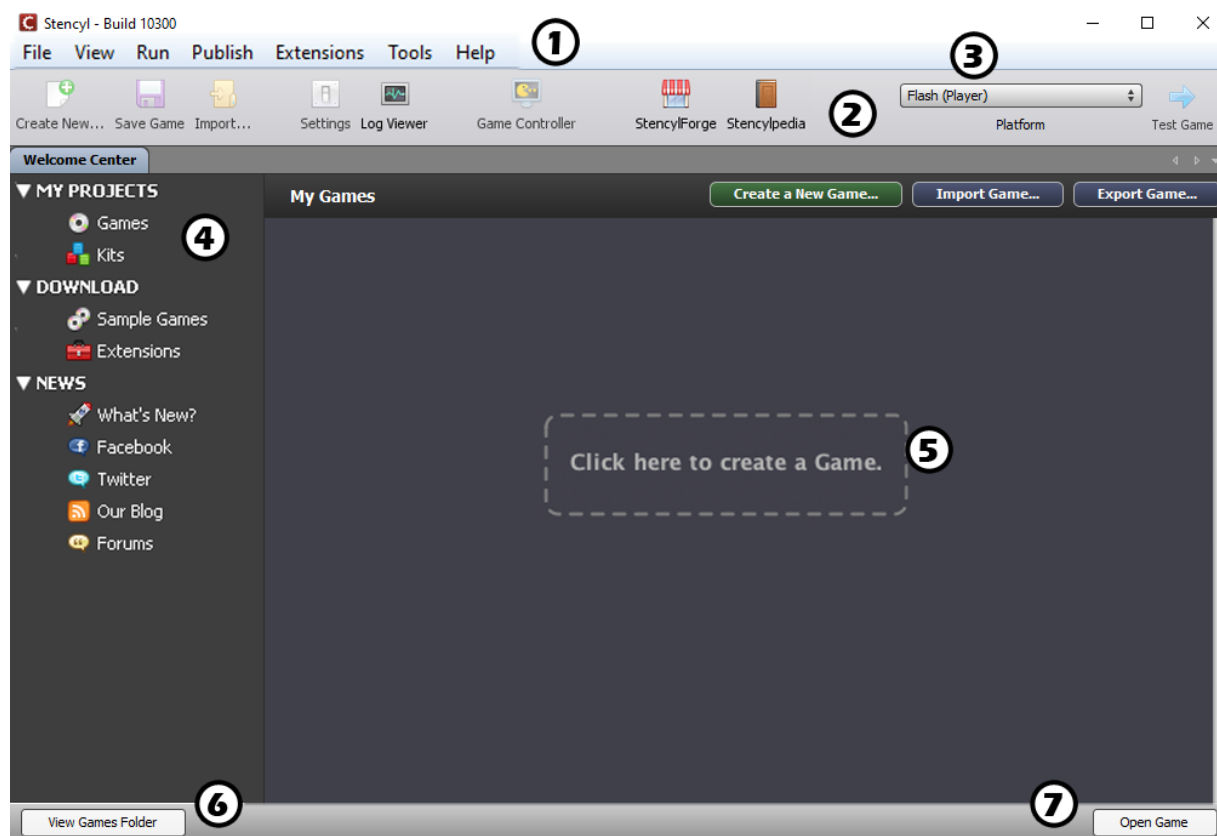


Figura 1.1: Tela de abertura do Stencyl

A interface do Stencyl é composta de:

1. Menu:

- *File* (criar, fechar, importar, exportar e atualizar os jogos);
 - *View* (visualizar e ocultar a barra de ferramentas e o StencylForge para baixar assets);
 - *Run* (testar os jogos);
 - *Publish* (opções de publicações);
 - *Extensions* (recursos extras para o software);
 - *Tools* (acesso a informações e recursos de instalação do software, como pastas e ajustes na instalação);
 - *Help* (acesso a tutoriais e informações adicionais).
2. *Toolbar* (barra de ferramentas): opções como criar, salvar, importar jogos, configurações e acesso a tutoriais e bibliotecas de arquivos do software.
 3. *Opções para testar o jogo*: as opções para o teste do jogo incluem as plataformas Flash, HTML5, iOS, Android, Mac e a linguagem Cppia.
 4. *Welcome Center*: aqui você tem acesso a seus jogos criados, exemplos e novidades do Stencyl.
 5. Clique nessa opção para iniciar a criação de um novo jogo.
 6. Visualização da pasta onde o jogo é armazenado quando é criado;
 7. Botão de abertura do jogo.

1.2 Criando seu primeiro jogo

Neste capítulo, iniciaremos a criação do nosso primeiro jogo que orientará todo o desenvolvimento do projeto até a finalização do livro: o "Jogo do Macaco", nome originado dos arquivos utilizados da biblioteca do software, *Monkey Game*. Durante a jornada do personagem, serão apresentados os principais recursos necessários para que a jogabilidade aconteça. Inclusão de arquivos relacionados

aos personagens e aos cenários, desenvolvimento de níveis, colisões entre os elementos incluídos no jogo, contagem regressiva, game over, enfim, tudo que um jogo necessita para se tornar "jogável".

O Stencyl, após ser instalado e aberto, não possui nenhum jogo. Siga os passos abaixo para a criação do primeiro jogo:

1. Clique na opção `Create a New Game` ;
2. Selecione a opção `Blank Game` para iniciar um jogo em branco e vá até a opção `Next` ;
3. Digite o nome **Jogo do Macaco**;
4. Clique no botão `Create` .

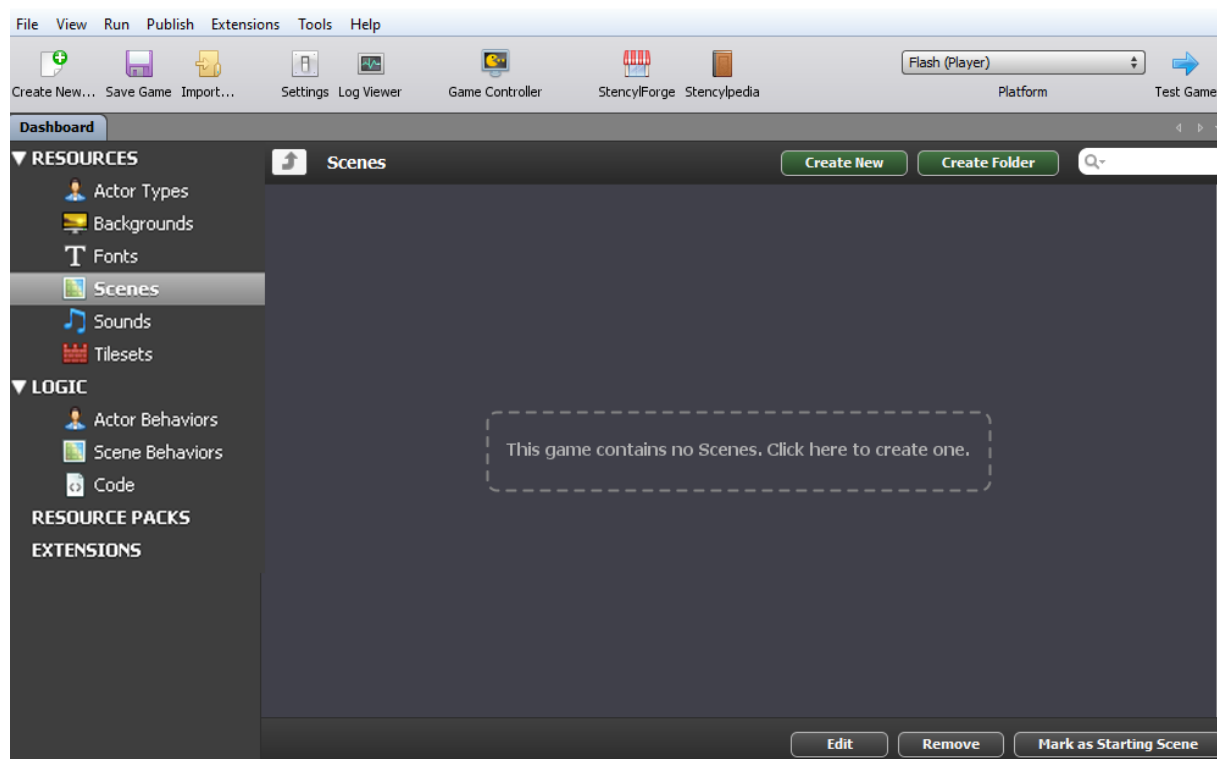


Figura 1.2: Tela de abertura do jogo

Repare que a tela inicial do jogo é semelhante à de abertura do Stencyl. A única grande diferença é um novo painel chamado *Dashboard* encontrado ao lado esquerdo da tela.

Dashboard é o local onde você encontrará todos os arquivos armazenados para o desenvolvimento do seu jogo. Personagens, backgrounds e arquivos de som importados serão encontrados neste painel. Veja cada um deles:

1. *Resources*:

- *Actor Types*: aqui estarão os arquivos referentes aos personagens utilizados no jogo;
- *Backgrounds*: telas de segundo plano para as cenas;
- *Fonts*: arquivos de fontes para o desenvolvimento das interfaces;
- *Sounds*: arquivos referentes aos efeitos de som e trilha sonora;
- *Tilesets*: blocos ou miniaturas utilizados para a construção da cena.

2. *Logic*:

- *Actor Behaviors*: comportamentos e eventos relacionados aos personagens;
- *Scene Behaviors*: comportamentos e eventos relacionados à cena.
- *Code*: inserção de instruções por códigos.

3. *Resource Packs* e *Extensions* são atualizações para o Stencyl.

1.3 Criando uma primeira cena (nível) para o jogo

Após clicarmos no botão criar, nosso jogo foi criado. No entanto, ele ainda não possui uma cena, então, o Stencyl nos posiciona automaticamente na opção *Scenes*. A cena é onde o jogo acontece, seu mundo, também conhecida como os níveis (*Levels*) do jogo. Para criarmos uma nova cena, vamos clicar na opção *Scenes*, dentro de *Resources* em *Dashboard*. Siga as etapas:

1. Clique na mensagem `This game contains no Scenes`. Click here to create one para criar uma nova cena;
2. Nomeie a cena como "Cena01";
3. Vamos manter por enquanto os valores pré-configurados para as dimensões no campo `size` ;
4. Em `Background Color` , escolha a cor "Sea green".
5. Clique em `Create` .

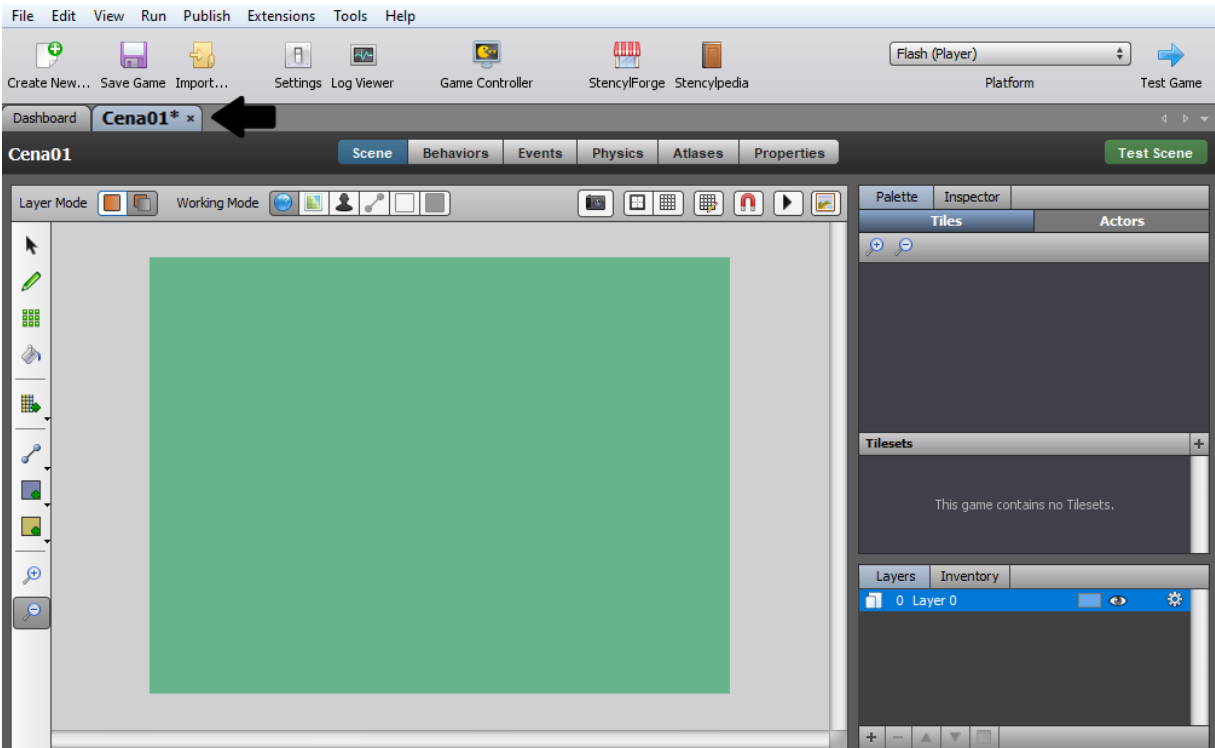


Figura 1.3: Tela relacionada à cena selecionada

Repare que, na parte superior esquerda da tela, uma nova guia foi criada, chamada `Cena01` . Clicando na guia *Dashboard* você retornará para a tela anterior.

As dimensões da cena, ou seja, sua largura e altura, podem ser visualizadas e reconfiguradas na guia *Properties*, acima da cena. Veja:

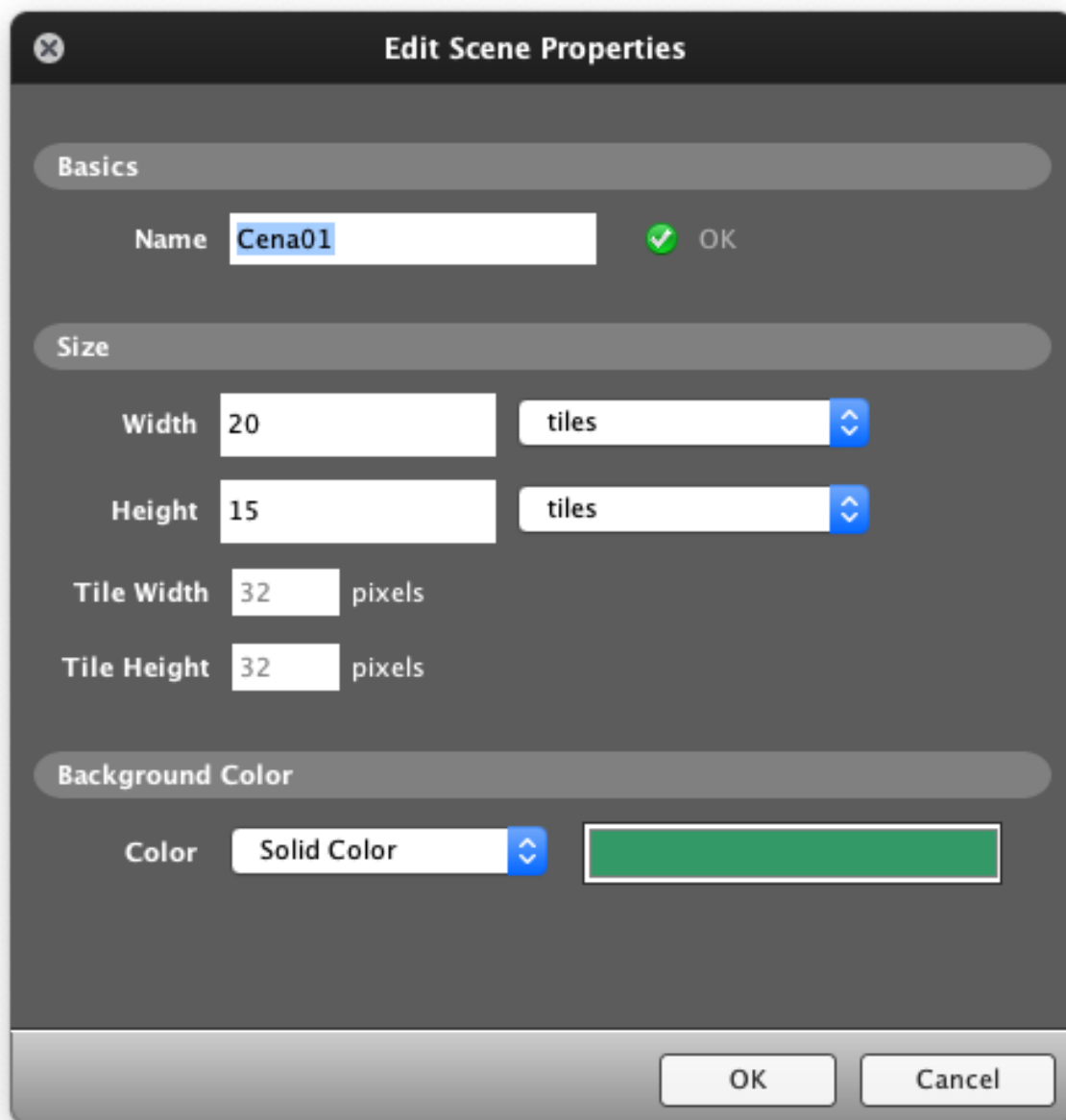


Figura 1.4: Janela referente às propriedades da cena

As dimensões são medidas em `Tiles`, pequenas miniaturas de 32 pixels, ou em pixels. Quando alteramos as dimensões, os novos valores são adicionados sempre à direita e abaixo da cena. Para nossa cena inicial, ou seja, nosso primeiro nível, utilizaremos 20 *tiles* de largura (640 pixels) e 15 de altura (480 pixels). Posteriormente, aprenderemos a editar essa configuração.

Cena e Display

Como vimos anteriormente, a cena está relacionada ao nível do seu jogo. O Display é a parte da cena que será visualizada na tela do dispositivo onde o jogo rodará. Visualizamos apenas uma parte da cena na tela do dispositivo e, conforme vamos avançando, a cena vai se desenrolando. Podemos conferir as dimensões do Display clicando em `Settings` na barra superior, `Settings` novamente e `Display`. O Stencyl já relaciona os valores do Display conforme as dimensões da cena criada, supondo que você desenvolverá seu jogo com as mesmas dimensões. Veja:

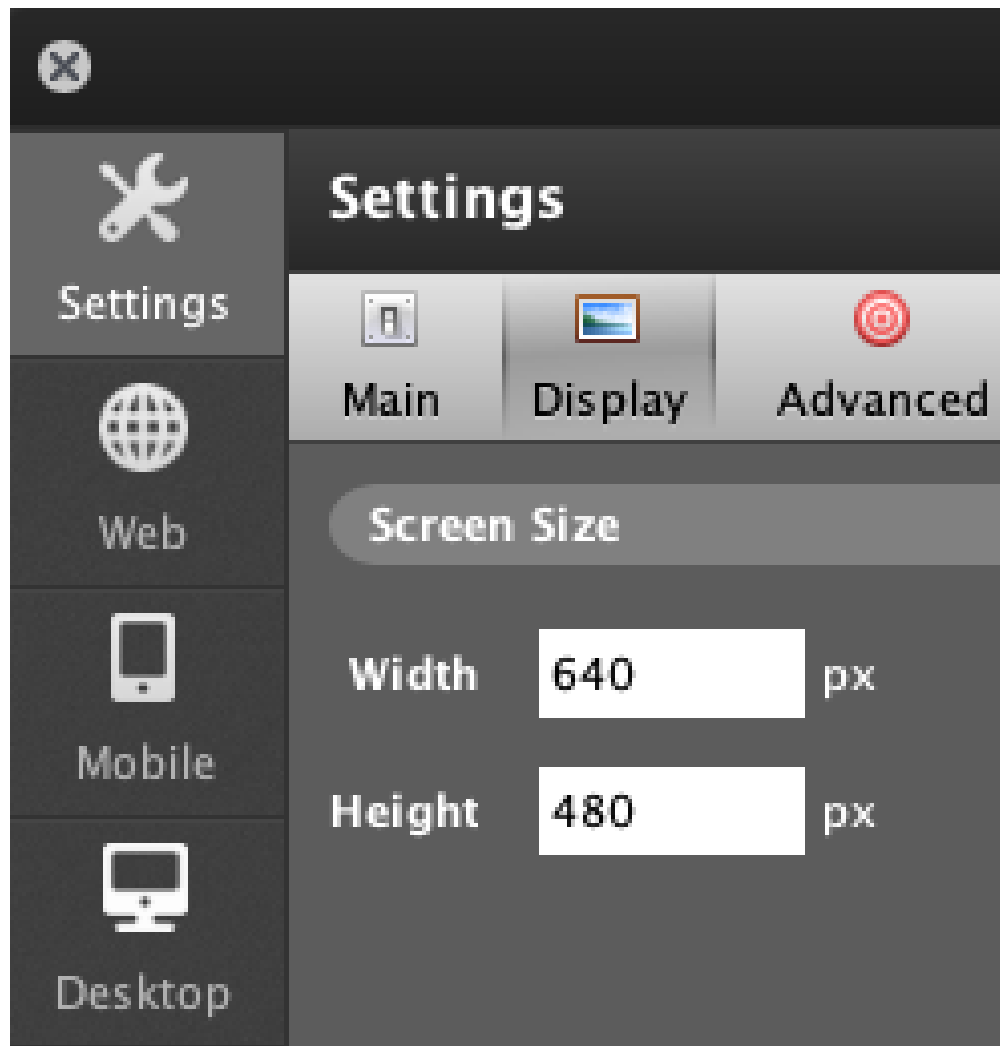


Figura 1.5: Janela referente à configuração da tela

Para guardar as alterações efetuadas até o momento, clique no botão `Save` no canto superior esquerdo da tela para salvar o jogo.

Lembre-se de que para acessar o arquivo do jogo você deve clicar em `View Games Folder`, na tela inicial do Stencyl. E para exportar para outro local, você deve utilizar a opção `Export Game`, encontrada no menu `File`.

1.4 StencylForge

StencylForge é onde encontraremos os arquivos para o desenvolvimento do nosso jogo. Personagens, *tiles*, *backgrounds*, enfim, tudo o que vai compor o cenário do jogo. Para podermos ter acesso aos arquivos do StencylForge, é necessário que você crie uma conta no software. Feche as guias que porventura estiverem abertas e siga as etapas adiante:

1. Clique em `File` desça até `Sign in` ;
2. Clique em `Create account` ;
3. Digite um nome para o campo `Username` ;
4. Crie uma senha no campo `Password` e repita no campo abaixo;
5. Digite seu email no último campo;
6. Clique em `Sign up` .

Pronto! Sua conta foi criada.

Baixando os arquivos do StencylForge

Veja abaixo os passos para baixar os arquivos necessários à realização das atividades:

1. Clique em **StencylForge** na barra de ferramentas;
2. Em `Resources` , clique em `Actor Types` ; Aqui você encontrará todos os personagens e elementos que servirão para compor seu jogo, cenário e instruções.
3. Selecione o *actor* **Stencyl Book Monkey** e clique em `Download` ; Esse será nosso personagem principal. É em torno dele que nossa jornada se desenrolará.
4. Quando o download terminar, aparecerá a tela de edição da animação. Veja:

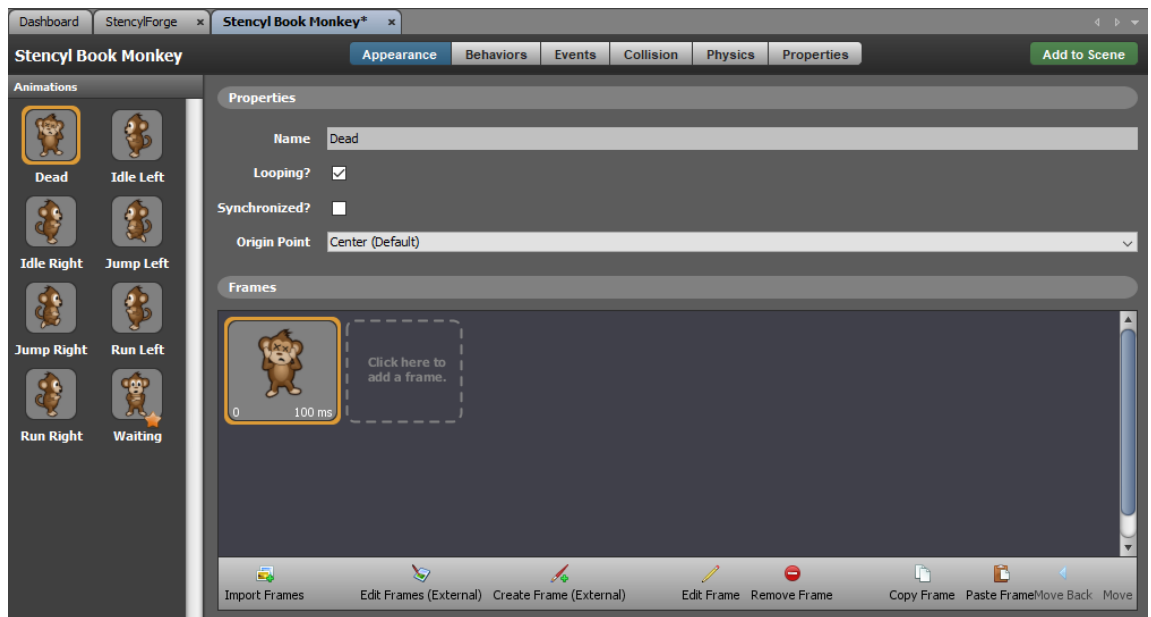


Figura 1.6: Tela referente ao actor selecionado

5. Feche a aba chamada `Stencyl Book Monkey` e clique na aba `StencylForge` ;
6. Volte para os outros *actors* clicando na seta acima do *actor* `Stencyl Book Monkey` :

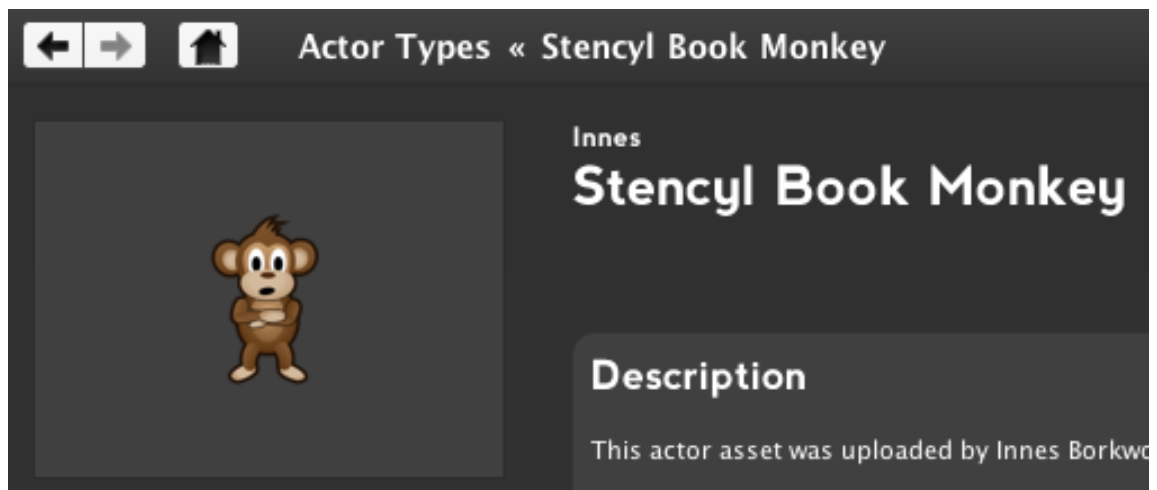


Figura 1.7: Destaque para a seta de retorno aos outros Actor Types do StencylForge

Baixe os personagens `Stencyl Book Croc` e `Stencyl Book Snake` . Cada personagem baixado será um ator no jogo. No nosso jogo, eles

serão os vilões, servirão de obstáculos para o nosso personagem Macaco, dificultando sua passagem pelos níveis. Além deles, posteriormente baixaremos outros arquivos para ir completando o nível.

Os arquivos que utilizaremos neste livro foram criados por Vicki Wenderlich, uma artista digital que disponibilizou gratuitamente suas ilustrações para serem utilizadas em artes digitais. Por estarem na biblioteca do Stencyl, sua utilização se torna acessível para todos aqueles que desejam aprender o software.

Inserindo um *actor* na cena

Já baixamos nossos personagens, agora vamos colocá-los em cena. Sempre que um *actor* é baixado do StencylForge, ele fica guardado em *Actor Types*, no painel `Dashboard`. Para adicionar um ator à cena, confirme se você está na aba do ator desejado. Caso contrário, selecione-o em *Actor Types* no painel `Dashboard`.

1. Clique duas vezes sobre o *actor* `Stencyl Book Monkey` ;
2. À direita clique em `Add to Scene` ;
3. Na tela `Choose a Scene` , selecione `Cena01` e clique em `OK` ;
4. Clique na cena para inserir o *actor*;
5. Pressione a tecla `ESCAPE` para cancelar o procedimento, ou seja, eliminar do cursor do mouse o último *actor* selecionado;
6. Salve o jogo.



Figura 1.8: Tela referente à inclusão do actor na cena

DICA:

Durante o desenvolvimento do jogo, é importante testá-lo para visualizar se os recursos inseridos estão funcionando corretamente. Para isso, basta escolher a plataforma onde o jogo rodará e, em seguida, clicar em *Test Game* no canto superior direito da tela.

Para testar nas plataformas iOS e Android, baixe os complementos Xcode e JDK (Java Development Kit). Veremos com mais detalhes no último capítulo do livro.

1.5 Tiles

Um *tile* é uma pequena imagem, uma miniatura que serve como partes para se realizar uma imagem maior para compor o cenário. Refere-se ao método de construção de mapas e níveis em um jogo, sendo colocados em ordem e posições para que preencha as áreas adequadamente. Precisaremos dos *tiles* para incrementar nossa cena e também inserir elementos e obstáculos para que nosso personagem possa interagir com eles.

Baixando tiles do StencylForge

Para desenvolvermos nossa primeira cena, utilizaremos algumas miniaturas que servirão de chão e plataforma para que nosso personagem se desloque. Primeiramente, vamos baixar o arquivo `Stencyl Book Tileset` no StencylForge e seguir as etapas:

1. Clique à esquerda da tela em `Tilesets` ;
2. Clique duas vezes na miniatura `Stencyl Book Tileset` . Aparecerá a seguinte imagem contendo um conjunto predefinido de *tiles* dentro de um editor de *tiles*:

Stencyl Book Tileset



| | A | B | C | D | E | F |
|----|---|---|---|---|---|---|
| 1 | | | | | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | | | | | | |
| 5 | | | | | | |
| 6 | | | | | | |
| 7 | | | | | | |
| 8 | | | | | | |
| 9 | | | | | | |
| 10 | | | | | | |
| 11 | | | | | | |
| 12 | | | | | | |
| 13 | | | | | | |
| 14 | | | | | | |
| 15 | | | | | | |
| 16 | | | | | | |
| 17 | | | | | | |



Figura 1.9: Tela referente ao Tiles baixado do StencylForge

Inserindo e trabalhando com tiles na cena

Para inserir os *tiles* na cena é muito simples:

1. Acesse a aba da cena criada;
2. No painel *Pallette* à direita da tela, clique em `Tiles`;



Figura 1.10: Painel referente ao Tile selecionado

3. Você encontrará neste painel os *tilas* descarregados. Para inseri-los, basta selecionar o *tile desejado* e clicar na cena. Veja

o exemplo a seguir (no próximo capítulo, concluiremos o cenário com mais miniaturas):

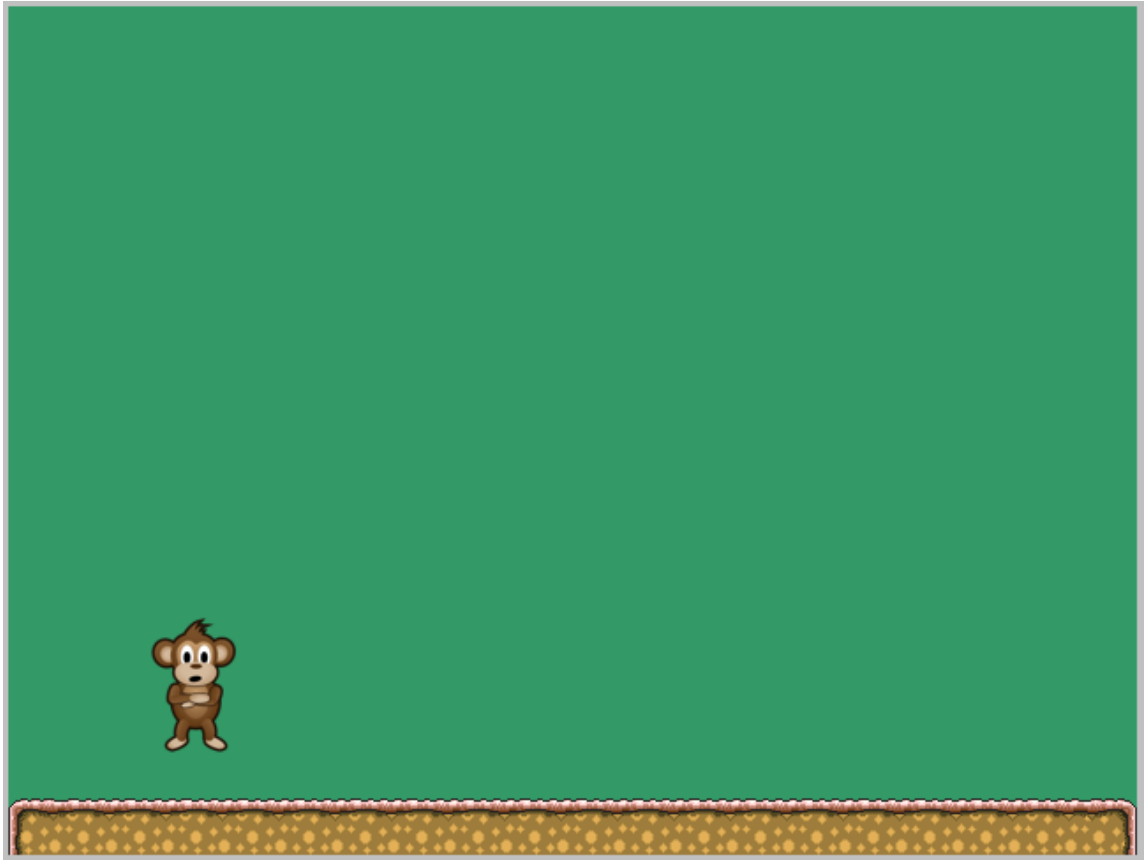


Figura 1.11: Tela com os tiles incluídos

4. Teste seu jogo.

DICAS:

- Para selecionar *tiles* ou *actors*, clique na ferramenta *Select* na barra de ferramentas encontrada à esquerda da cena;
- Caso queira inserir várias cópias de um mesmo *tile*, basta clicar mantendo o botão do mouse pressionado;
- Para movê-lo, basta clicar e arrastar o *tile* para um outro local com a ferramenta *Select*;
- Para substituir um *tile*, basta clicar sobre o anterior com o novo selecionado;
- Para deletar algum *tile*, basta selecioná-lo e pressionar a tecla `delete` do teclado.

Neste capítulo:

- aprendemos como criar nosso primeiro jogo;
- aprendemos como criar cenas (níveis) e inserir *actors* e *tiles* pelo StencylForge.

No próximo capítulo, aprenderemos a movimentar nosso personagem pela cena e também vamos inserir novos *actors* e *tiles* para compor nosso primeiro nível. Até lá!

CAPÍTULO 2

Utilizando behaviors no jogo

Para que os arquivos incluídos no jogo se comportem adequadamente, precisamos inserir algumas instruções. Quando um certo evento ocorre, podemos definir o que acontecerá com um determinado personagem ou objeto no jogo. Chamamos a esses comportamentos de *behaviors*, uma reação a um evento. Por exemplo, quando pressionarmos uma determinada tecla, como o personagem vai reagir? Quando colidir com um obstáculo, o que acontecerá? São esses eventos que determinarão os *behaviors*, ou seja, os comportamentos dos personagens.

Neste capítulo, aprenderemos a inserir e configurar vários *behaviors* para o nosso jogo. Dessa forma, já poderemos nos divertir um pouco.

2.1 Inserindo e configurando behaviors

O Stencyl já traz alguns *behaviors* pré-configurados, assim já podemos utilizá-los para nos familiarizarmos com eles. Nos próximos capítulos nos aprofundaremos mais na criação e no desenvolvimento de *behaviors*, mas por enquanto vamos utilizar alguns *behaviors* preexistentes para dar uma movimentação básica ao nosso *actor* já inserido na cena. Caso a aba do nosso personagem `Stencyl Book Monkey` esteja fechada, clique duas vezes nele em `Actor Types` no painel `Dashboard`:

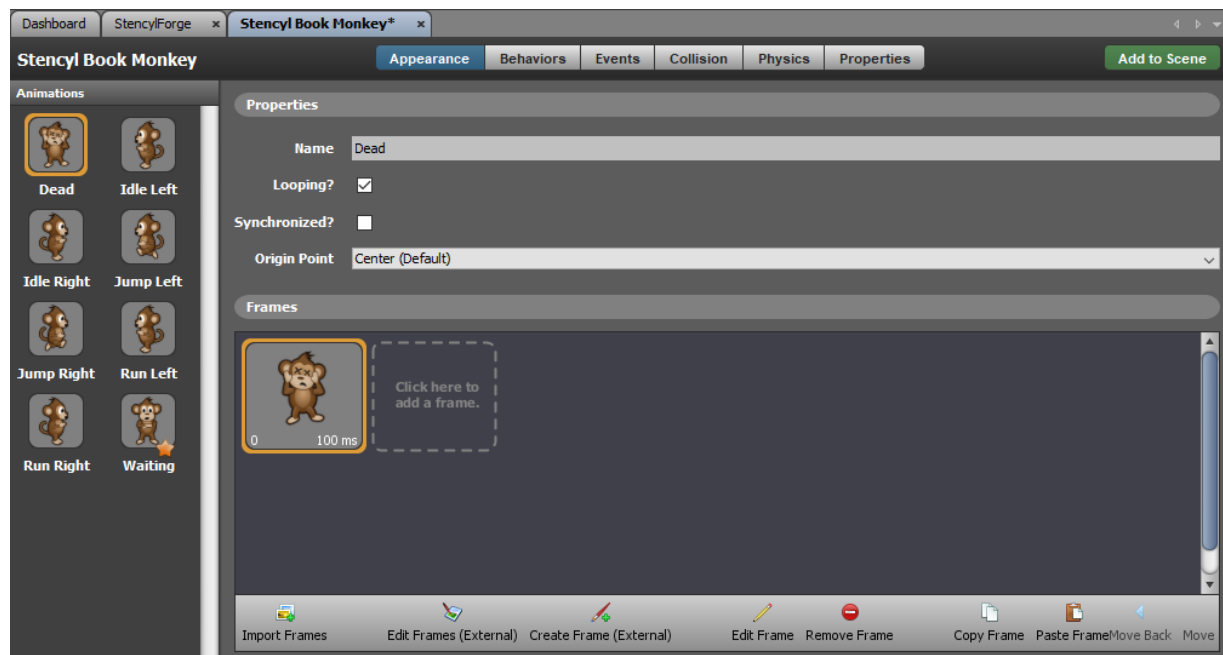



Figura 2.1: Tela de animações do actor

1. Com a aba `Stencyl Book Monkey` exibida, clique na guia `Behaviors` .
Aqui você acessará a biblioteca de *Behaviors do software*.
2. Clique na mensagem `Click here to choose... Actor Type` ;
3. Na janela `Choose a Behavior` , você encontrará vários *behaviors* divididos por categorias:
 - *Motion*: *behaviors* relacionados ao descolamento do *actor* na cena.
 - *Game*: instruções para o funcionamento de botões e utilização de câmera para o *actor*.
 - *Controls*: opções para o controle adequado, via teclado ou mouse, por exemplo, dos *actors* no jogo.
 - *Utilities*: alguns utilitários para o jogo, como links para sites e redes sociais.
 - *Eye Candy*: *behaviors* para efeitos visuais.
 - *GUI*: mais opções de efeitos visuais para controles e botões.
 - *Rules*: alguns *behaviors* para quando o *actor* sair de cena.

Para movimentarmos nosso *actor* pela cena, utilizaremos o `behavior` `Jump and Run Movement` .

1. Clique em `Jump and Run Movement` encontrado dentro da categoria `Controls`.

Esse *behavior* dará movimento ao nosso personagem. Ele poderá se movimentar para o lado, pular e, além disso, poderemos vincular aos movimentos as animações do *actor*. Na janela de configuração do *behavior*, precisamos inserir alguns atributos para que o personagem possa se movimentar. Utilizaremos as setas do teclado. Vamos fazer as alterações? Veja a imagem:



Jump and Run Movement

Allows the actor to run and jump while on the ground.

Attributes

| | | |
|------------------------|-------------------------------------|--|
| Use Controls | <input checked="" type="checkbox"/> | <p>If selected, the control attributes will be used to control If unselected, the control attributes are ignored and the actor move as if the keys were pressed.</p> |
| Left Control | <input type="text" value="left"/> | |
| Right Control | <input type="text" value="right"/> | |
| Jump Control | <input type="text" value="up"/> | |
| Running Force | <input type="text" value="50"/> | <input checked="" type="checkbox"/> <p>The force used for running.</p> |
| Maximum Running Speed | <input type="text" value="15.0"/> | <input checked="" type="checkbox"/> <p>The maximum speed at which the actor can run.</p> |
| Jumping Force | <input type="text" value="25.0"/> | <input checked="" type="checkbox"/> <p>The force used for jumping.</p> |
| Variable Jump | <input type="checkbox"/> | <p>If selected, the jumping height will depend on how long</p> |
| Variable Jump Duration | <input type="text" value="0.2"/> | <input checked="" type="checkbox"/> <p>Holding down the jump key longer than this will not mak</p> |

Figura 2.2: Atributos do behavior Jump and Run Movement

As opções a seguir referem-se a:

- *Left, Right and Jump Control*: controles para deslocamento à esquerda (*Left*), à direita (*Right*) e para o pulo (*Jump*); utilizaremos *left* para *left control* para que quando controlarmos nosso personagem pelo dispositivo ele se movimente para a esquerda; *right* para que ele se desloque para a direita; e *up*, para que, quando apertamos a seta do teclado para cima, ele pule.
- *Running Force*: valor relacionado à força utilizada para o *actor* correr. Utilizaremos aqui o valor 50. As alterações nessa opção são muito sutis. Valores muito menores fazem com que o *actor* não saia do lugar;
- *Maximum Running Speed*: valor relacionado à velocidade com que o *actor* pode correr. Quanto maior o valor, mais rápido nosso *actor* vai correr. Experimente o valor 15;
- *Jumping Force*: valor relacionado à força utilizada para o *actor* pular. Quanto maior o valor, maior o pulo. Vamos inserir aqui o valor 25.0 para que nosso *actor* não pule tão alto mas também não fique tão preso ao chão;
- *Variable Jump*: ative esta opção caso queira que o pulo do *actor* esteja relacionado ao tempo em que a tecla de controle estiver sendo pressionada;
- *Variable Jump Duration*: tempo relacionado à pressão da tecla de pulo. Só funcionará se a opção acima estiver selecionada.

DICA:

Experimente os valores nas opções anteriores. Dessa forma, você compreenderá melhor cada opção e adequará os melhores valores a mecânica do seu jogo.

Após a configuração, seguiremos nesta mesma janela para definir as animações à movimentação do *actor*.

Vinculando animações à movimentação do actor

Durante o desenvolvimento de personagens para os jogos, precisamos definir suas animações conforme se movimentam pelo mundo do jogo. Chamamos essas animações de *sprites*. *Sprites* são as variações de poses chaves de movimentação do personagem de modo a enfatizar gestos e atitudes de um personagem animado. No início deste capítulo, na imagem referente à tela de animações do *actor*, vimos que nosso personagem foi desenvolvido com 08 *sprites*, incluindo movimentações para a esquerda e direita, pulo e até quando ele morrer no jogo.

Com as animações já pré-desenvolvidas, precisaremos vinculá-las ao controle durante a jogabilidade, ou seja, quando apertarmos uma tecla ou acionarmos um comando para que ele se movimente para a esquerda, queremos que seja exibida a animação correspondente. Após ter rolado a tela de atributos do behavior Jump and Run Movement, configure os movimentos conforme a imagem a seguir clicando na opção Choose Animation... :

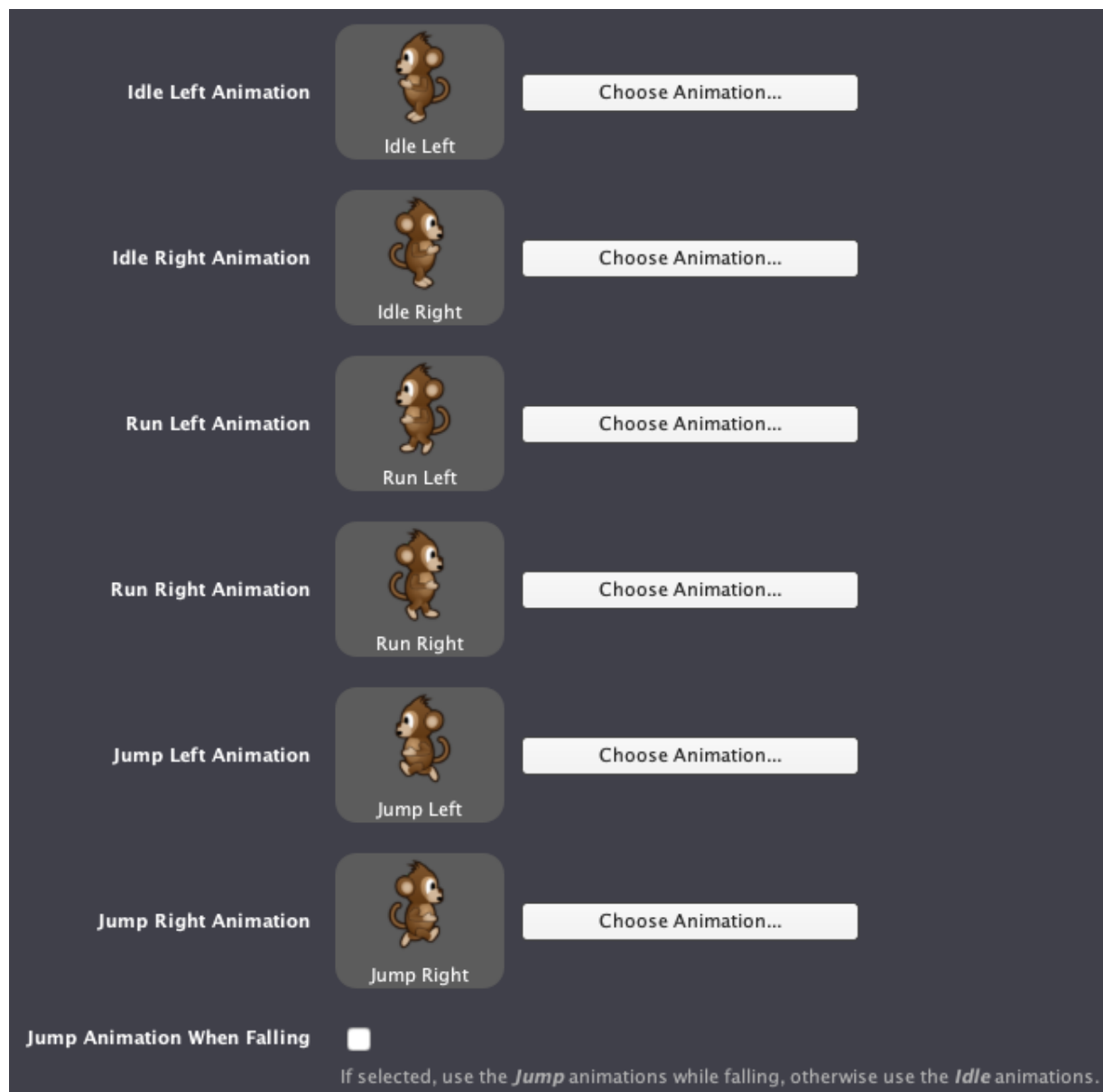


Figura 2.3: Tela de animações do behavior Jump and Run Movement

A opção `Jump Animation Falling` ativada relacionará as animações anteriormente definidas em `Jump Left` e `Jump Right` animations durante o tempo de queda do *actor*. Caso contrário, as animações atribuídas à queda serão as definidas em `Idle Left` e `Idle Right` animations .

Após salvar e testar o jogo, um pequeno problema aconteceu, apenas os movimentos à esquerda e à direita funcionaram e o

macaquinho está flutuando na tela. Para corrigir esse pequeno problema precisaremos inserir o efeito de gravidade na cena.

2.2 Inserindo o efeito de gravidade na cena

O recurso de gravidade é inserido na cena para que o *actor* não fique deslizando ou flutuando e pode ser adicionado tanto na horizontal (left/esquerda ou right/direita) como na vertical (up/para cima ou down/para baixo). Quanto maior o valor utilizado, mais presos ao chão os *actors* ficarão, fazendo com que saltem com mais dificuldade. Vamos compreender melhor seguindo as próximas etapas:

1. Com a Cena01 aberta, clique na guia Physics ;



Figura 2.4: Opções da guia Physics para a gravidade

2. Repare, como dito anteriormente, que podemos configurar a gravidade tanto na horizontal como na vertical. Configuraremos apenas a gravidade vertical. Experimente valores diferentes! Valores menores proporcionam saltos mais altos. Zerar a

gravidade impede que o *actor* se movimente. Vamos utilizar no nosso jogo o valor 90 em *Down*;

3. Teste o jogo.

Agora você perceberá que os atributos do *behavior* estão funcionando, mas alguns ajustes ainda são necessários. Vamos impedir que o nosso *actor* se movimente para fora da cena?

2.3 Behavior para o actor não sair de cena

Tendo como referência as etapas anteriores de como utilizar *behaviors* no jogo, tente manter o personagem na cena realizando os seguintes tópicos:

1. Clique duas vezes sobre o *actor* `Stencyl Book Monkey` ;
2. Na guia `Behaviors` , clique em `+ Add Behavior` , no canto inferior esquerdo da tela;
3. Dessa vez, selecionaremos a opção `Motion` na biblioteca de *behaviors*;
4. Clique duas vezes em `Cannot Exit Scene` ;
5. Salve e teste o jogo.

Movimente o *actor* para os limites da cena. Você verá que ele não desaparece mais. Fácil, não é? Vamos agora fazer um pequeno ajuste para tornar o jogo mais interessante?

DICA:

Cuidado para não confundir o `behavior Cannot Exit Scene` com o `Cannot Exit Screen`. *Scene* se trata da cena e *Screen* tela, ok?; Você pode também utilizar superfícies de colisão (ferramenta *Terrain*) para impedir que o *actor* saia de cena. A ferramenta *Terrain* cria espaços de colisões na cena que são invisíveis no jogo. Para utilizá-la, selecione-a na barra de ferramentas e arraste na área desejada na cena. Veja:



Figura 2.5: Área criada com a ferramenta Terrain

Para voltar à visualização normal, pressione o número **3** do teclado.

2.4 Reconfigurando a cena do jogo

Para tornar nosso jogo mais interessante, precisaremos inserir mais obstáculos e dificultar um pouco mais o nível. Para isso, vamos ampliar a cena do jogo e, em seguida, faremos a tela rolar para visualizar o restante da cena. Como vimos anteriormente, tela e cena são coisas diferentes: a cena é o tamanho do cenário onde o personagem vai circular, e a tela está relacionada ao dispositivo em que o jogo vai rodar. Vamos agora reconfigurar as dimensões da cena:

1. Selecione a `Cena01` no Dashboard clicando duas vezes sobre ela;
2. Clique na guia `Properties` ;
3. As dimensões são medidas em *tiles* ou pixels, sendo o tamanho padrão de um *tile* de 32x32 pixels como padrão. Uma largura de 60 *tiles* e altura de 15 equivalem a 1920 por 480 pixels. Utilizaremos neste livro o valor de 80 por 15 *tiles*, ou 2560 por 480 pixels;

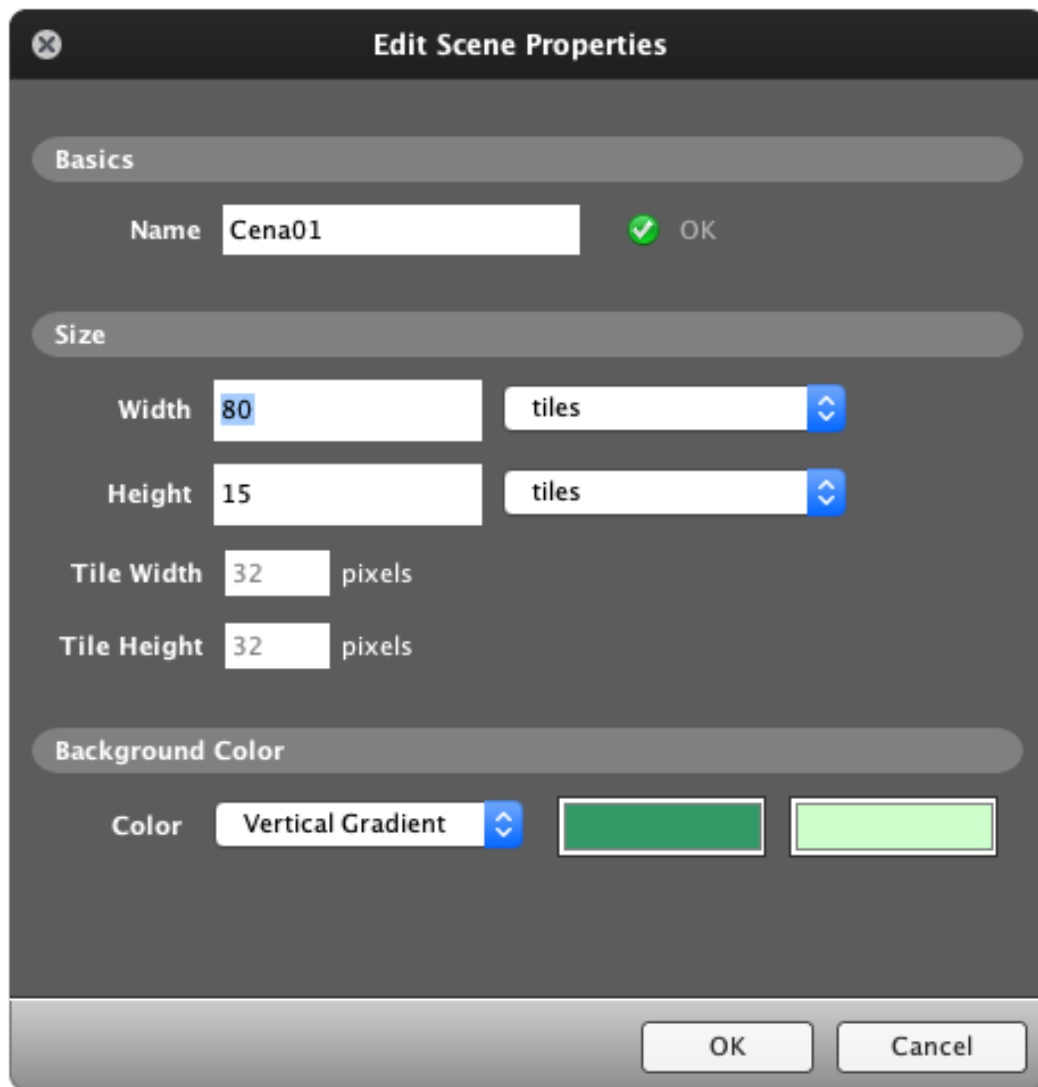


Figura 2.6: Propriedades da cena

4. Na parte de baixo da tela, na área *background*, você pode definir uma nova cor ou até transições tonais entre duas cores. Escolha livremente as cores que desejar;



Figura 2.7: Opções de gradiente da tela de propriedades da cena

5. Salve e teste o jogo.

Faça você mesmo!

Agora é a sua vez. Preencha toda a cena com os *tiles* faltantes e outros *actors* para compor a cena. Aproveite para inserir novos *actors*: Stencyl Book Croc , Stencyl Book Snake e as frutas Stencyl Book Bananas , Stencyl Grapes , Stencyl Melon e Stencyl Book Pineapple (não esqueça de baixá-los no StencylForge). Os *actors* Stencyl Book Croc e Stencyl Book Snake serão os obstáculos para o avanço do nosso *actor* principal, e os *actors* Bananas , Grapes e Melon serão as frutas que nosso *actor* precisa coletar para avançar para o próximo nível. Insira esses novos elementos de forma que as dificuldades aumentem quantitativamente (um aumento na quantidade de elementos que sirvam como obstáculos, por exemplo. Você pode inserir o mesmo *actor* mais de uma vez na cena de forma que o nível de dificuldade vá aumentando conforme o macaquinho avance na cena) e qualitativamente (um aumento qualitativo no nível de dificuldade, como aumentar a distância entre os *tiles* que sirvam de plataforma para que nosso *actor* tenha dificuldade em saltar entre elas). Planejar os níveis de dificuldade é importante para que o jogador se sinta motivado a aumentar sua habilidade técnica para superar os níveis mais difíceis. Caso contrário, ele pode se sentir entediado no jogo. Veja o exemplo:

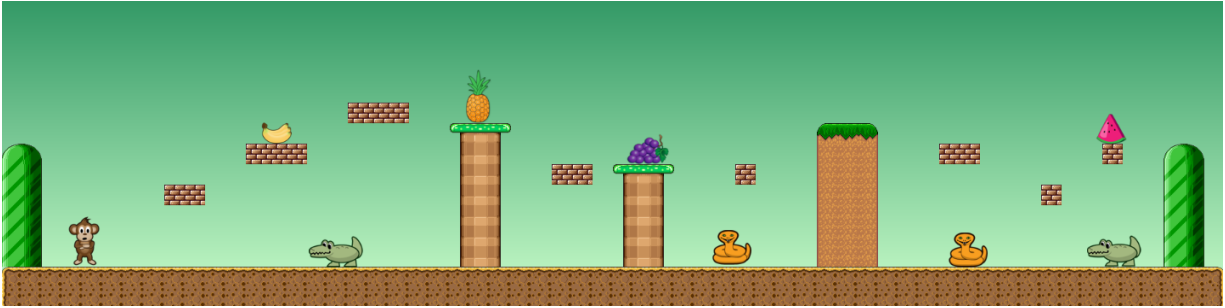


Figura 2.8: Exemplo de cena para o primeiro nível


DICA:

Para facilitar a inclusão dos *tiles*, utilize a opção de *grids* pressionando a letra **g** do teclado.

2.5 Ajustes gerais para os actors

Após ter incluído outros *actors* na cena, alguns ajustes são fundamentais para que eles se comportem adequadamente durante o jogo. Vamos conhecer cada um deles separadamente:


1. Primeiramente, abra o *actor* Stencyl Book Banana ;
2. Clique na guia *Physics* e, em seguida, em *General* . Aqui nós configuraremos alguns aspectos gerais sobre a física do personagem:



What kind of Actor Type?

Determines whether the Actor Type can move or not.

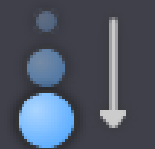
- ☐ Cannot Move
- ☐ Cannot be pushed (e.g. platforms)
- ☒ Normal



Can Rotate?

Determines if this Actor Type can rotate freely.

- ☐ No
- ☒ Yes



Affected by Gravity?

Gravity exerts a force on all actors in a scene.

- ☐ No
- ☒ Yes

Figura 2.9: Janela para a configuração de aspectos da física do actor

3. *What Kind of Actor Type*: aqui nós podemos impedir a movimentação do *actor* (*Cannot Move* : essa opção desabilita toda a configuração abaixo), impedir que ele seja empurrado (*Cannot be pushed*) ou permitir sua movimentação (*Normal*).
4. *Can Rotate?*: podemos permitir que rotacione quando se movimenta;
5. *Affected by Gravity*: anteriormente definimos a gravidade para a cena. Nessa opção podemos definir se essa gravidade definida

para a cena afetar um determinado *actor* ou não. Caso desabilite, mesmo com a gravidade na cena, os *actors* poderão flutuar.

Vamos habilitar a opção `Cannot Move` para que seja assumida a gravidade da cena para o `Stencyl Book Monkey`. Faça a mesma alteração para as outras frutas.

2.6 Rolando a tela

Após ter inserido os novos *tiles* e *actors*, quando for movimentar o personagem, a tela ainda não está acompanhando sua movimentação, ou seja, o personagem some. Para fazermos a tela rolar acompanhando a movimentação do personagem, vamos seguir as etapas:

1. Selecione o personagem desejado. Neste caso, `Stencyl Book Monkey`;
2. Clique na guia `Behaviors`;
3. Vamos em `+Add Behavior` no canto inferior esquerdo da tela para inserir um novo *behavior*;
4. Na opção `Game`, dê um clique duplo em `Camera Follow`. Esse *behavior* fará com que a câmera do dispositivo acompanhe a movimentação do *actor*. Quanto ele chegar na metade da câmera, ela se movimenta acompanhando-o e segue assim até o fim da cena;
5. Mantenha o valor padrão `0.0` para `Scroll Speed` para que a rolagem da câmera esteja adequada à movimentação do *actor*;
6. Salve o jogo.

Quando for testá-lo, você perceberá que, quando o personagem chega no meio da tela, ela começa a rolar enquanto ele se movimenta na cena, e assim permanecerá até que o personagem atinja o limite final da cena.

DICA:

Caso precise desabilitar um determinado *behavior* criado para um *actor*, selecione-o na guia Behavior da tela do *actor* e clique em Desactivate Behavior, no canto superior direito da tela.

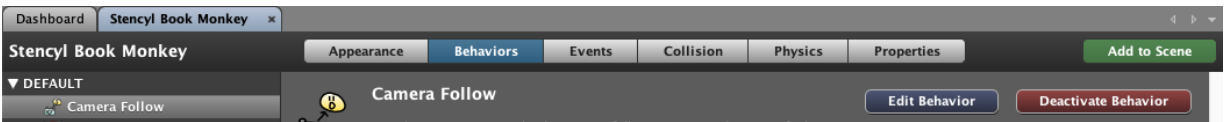


Figura 2.10: Opção para a desativação de behavior

Neste capítulo:

- aprendemos a utilizar os *behaviors* do Stencyl;
- inserimos efeito de gravidade na cena.

No próximo capítulo compreenderemos a importância dos shapes de colisão e como editá-los.

CAPÍTULO 3

Trabalhando com colisões

Para que o personagem possa detectar adequadamente os objetos ou outros *actors* que aparecem durante o jogo, sejam *actors* ou *tiles*, faz-se necessário editar seus *shapes* de colisão. Os *shapes* de colisão são formas que devem se adequar ao desenho dos arquivos inseridos no jogo, e editá-los é fundamental para a visualização correta do contato entre eles. Para compreendermos melhor, veja como estão os *shapes* de colisão do *actor* Stency1 Book Monkey :



Figura 3.1: Visualização de um **shape** de colisão

Perceba que a forma retangular do *shape* não está adequada ao desenho do nosso macaquinho.

Para podermos visualizar os *shapes*, vamos primeiramente habilitar o recurso `Enable Debug Drawing`, localizado no menu `Run`. Este recurso auxilia a visualização de como as áreas de colisão estão configuradas no nosso jogo. Para visualizar os *shapes*, teste o jogo. Repare que, durante o teste, aparecem algumas caixas que demonstram os limites de colisão nos elementos inseridos no jogo.



Figura 3.2: Visualização dos shapes de colisões com a opção `Enable Debug Drawing` selecionada

Veja que os *shapes* de colisão para os *actors* são mostrados com cores diferentes (um tom de vermelho ou roxo), e para os *tiles* com a cor verde. Podemos reparar também que esses *shapes* não estão adequados ao desenho dos elementos, como no exemplo do *actor* `Stencyl Book Monkey` e também nos bloquinhos amarelos.

3.1 Editando os shapes de colisões

Percebemos na imagem anterior que alguns *shapes* de colisão não estão adequados aos seus respectivos *actors* ou *tiles*. Vamos editá-los? Primeiramente, vamos ajustar os *shapes* de colisão para o nosso herói:

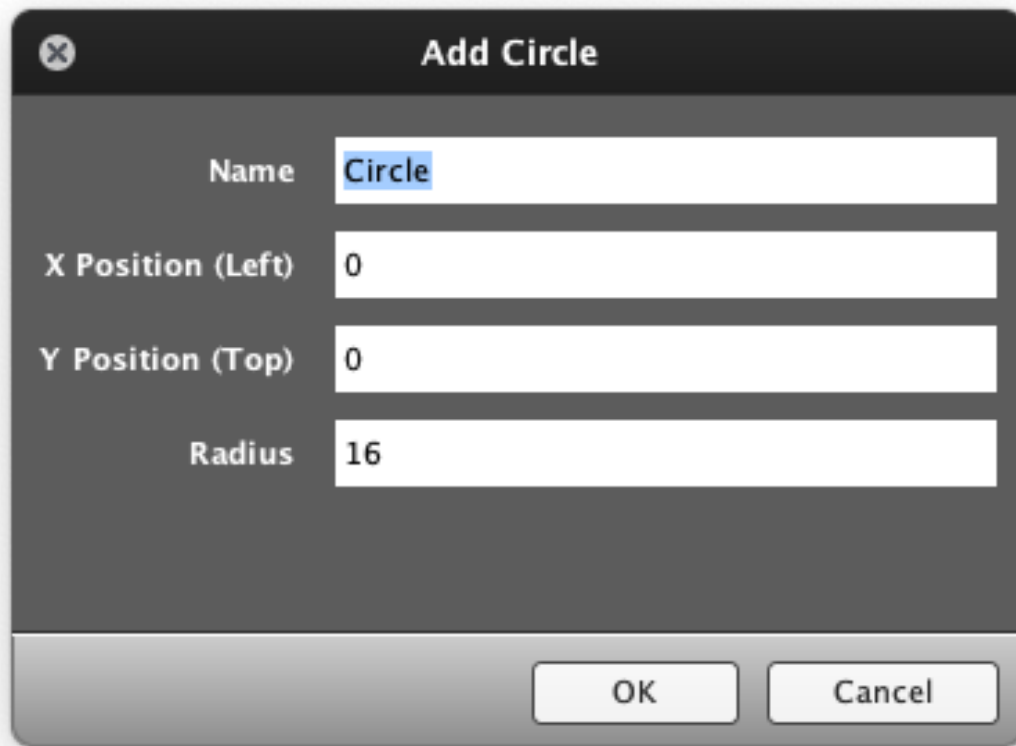
1. Selecione o personagem em `Actor Types` no painel `Dashboard` ;
2. Clique na guia `Collision` ;
3. No painel de animações, selecione `Idle Left` . Essa animação refere-se ao movimento do nosso *actor* para a esquerda;

Veja que, na parte superior da tela, podemos visualizar vários shapes que poderão ser utilizados de acordo com a forma de nosso *tile* ou *actor*.



Figura 3.3: Tipos de shapes para a edição das colisões

1. Clique sobre o *shape* do nosso personagem para excluí-lo. Pressione a tecla `delete` ;
2. Desta vez, vamos utilizar o *shape circular*, pois nosso *actor* possui formas arredondadas. Clique em `Add Circle` ;

A dialog box titled "Add Circle" with a close button (X) in the top-left corner. It contains four input fields: "Name" with the value "Circle", "X Position (Left)" with the value "0", "Y Position (Top)" with the value "0", and "Radius" with the value "16". At the bottom right, there are two buttons: "OK" and "Cancel".

| Field | Value |
|-------------------|--------|
| Name | Circle |
| X Position (Left) | 0 |
| Y Position (Top) | 0 |
| Radius | 16 |

Figura 3.4: Opções da janela Add Circle

3. As opções acima referem-se ao nome do *shape*, seu deslocamento na horizontal (*X Position (Left)*), o deslocamento na vertical (*Y Position (Top)*), e o ajuste da circunferência (*Radius*). Como ainda não sabemos exatamente a forma adequada do *shape* ao desenho, vamos confirmá-los clicando em `OK` e editá-los na próxima etapa;
4. Após clicarmos em `OK`, visualizamos a mesmas opções da tela anterior agora do lado direito. Utilize os valores do exemplo a seguir para editar o *shape*. Utilizando o mouse, podemos apenas movê-lo; para modificar seu tamanho, devemos apenas utilizar as opções da janela:

| Current Shape | |
|---------------------|--------------------------|
| X Position (Left) | 17 |
| Y Position (Top) | 9 |
| Radius | 18 |
| Physical Properties | |
| Is a Sensor? | <input type="checkbox"/> |

Figura 3.5: Janela de edição de shape de colisão

5. Pronto! O *shape* de colisão da cabeça do personagem ficou adequado à sua forma.



Figura 3.6: Visualização da edição do shape

6. Salve e teste o jogo.

Editando shapes de colisão poligonais

A utilização dos *shapes* de colisão poligonais é mais adequada para aqueles *actors* ou *tiles* que não possuem formas elípticas ou retangulares. Veja como é fácil utilizá-los:

1. No painel `Dashboard` , em `Actor Types` , abra o *actor Stencyl Book Melon*;
2. Clique na guia `Collision` ;
3. Clique em `Add Collision` ;
4. Veja um exemplo da edição de um *shape* poligonal. Para mover os pontos basta clicar sobre eles; para acrescentar mais pontos, clique no sinal de `+` ; para remover, clique no sinal de `-` . Veja o resultado após a edição:

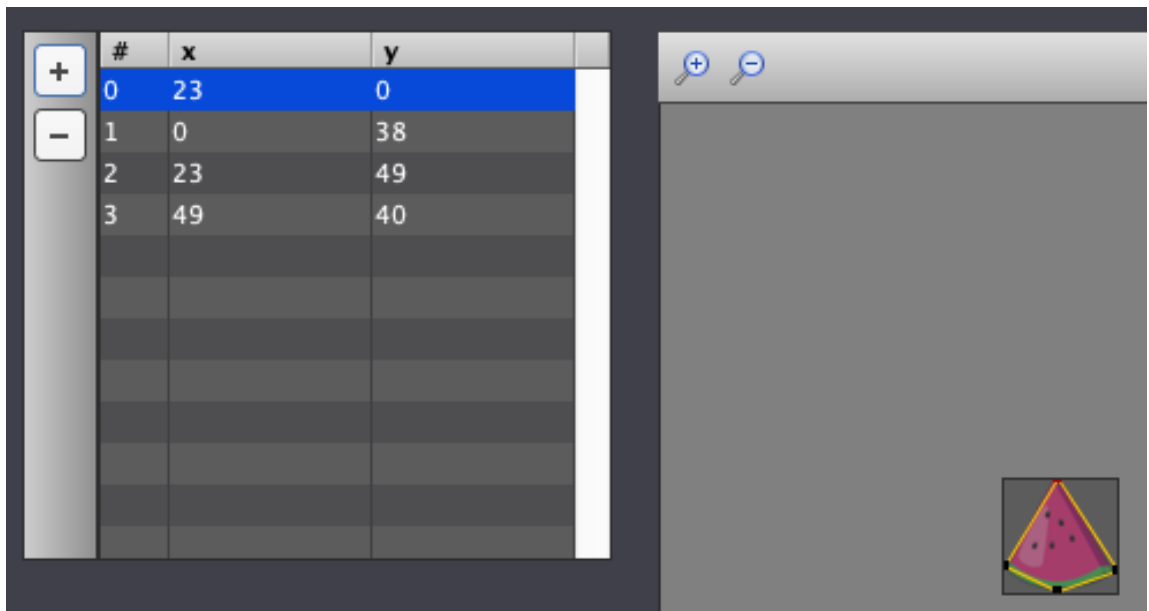


Figura 3.7: Visualização de um tile editado

5. Clique em **OK** e teste o jogo.

Faça você mesmo!

Viu como é fácil? Agora é a sua vez. Modifique os outros *shapes* de colisão para esta animação. Depois, modifique também os *shapes* para as outras animações deste *actor*. Modifique apenas os *shapes* das animações que sofrerão colisões, para que não tenha um trabalho desnecessário. No nosso jogo, as animações *Dead* e *Waiting* não sofrerão colisões.

Edite também as colisões dos outros *actors* inseridos na cena: o jacaré, a cobra e todas as outras frutas. Utilize as etapas aprendidas anteriormente. Não esqueça de sempre testar o jogo para visualizar as edições.

3.2 Utilizando shapes de colisão para os tiles

O processo de edição de colisões para os *tiles* não difere muito do procedimento realizado para os *actors*. Editaremos os *tiles* visualizados a seguir. Veja no teste do jogo como estão os *shapes* de colisão nesses *tiles* mostrados por meio das bordas verdes:

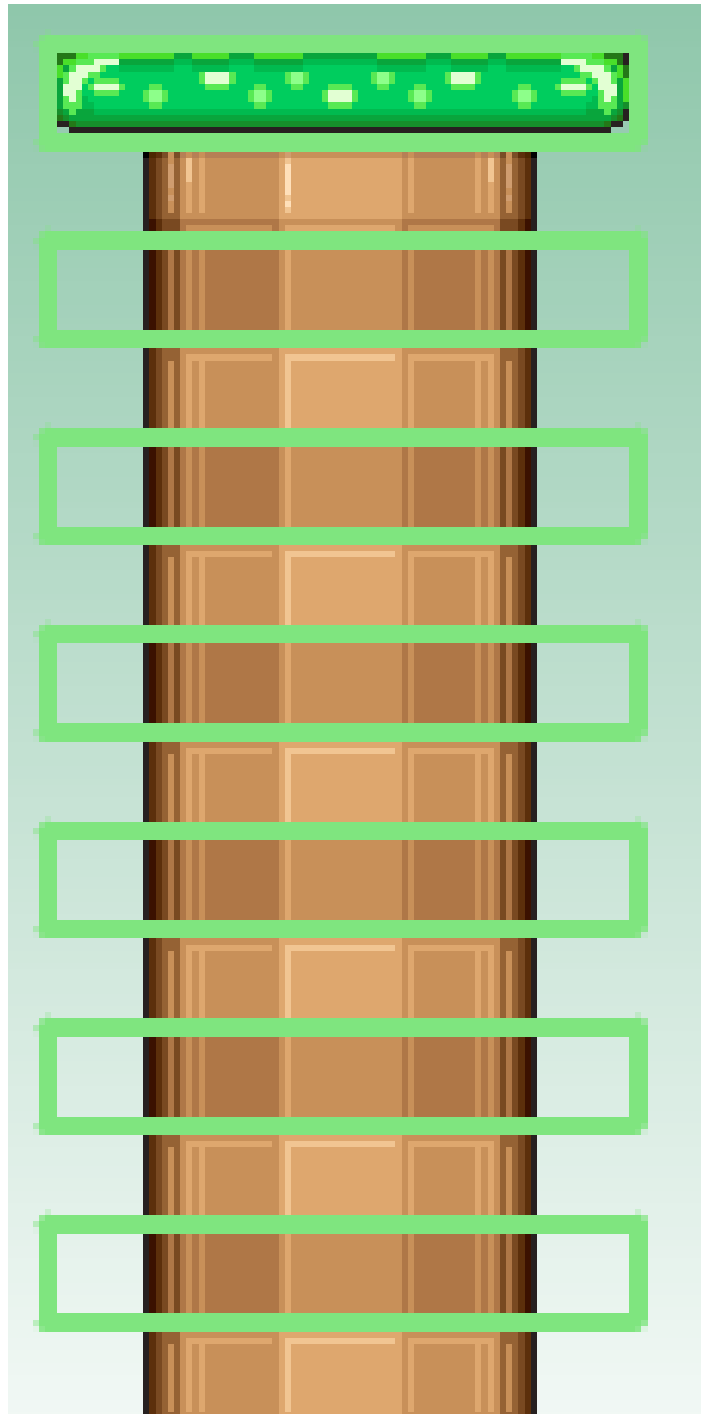


Figura 3.8: Tiles durante a visualização do jogo sem a edição dos shapes

1. Selecione o grupo de *tiles* `Stencyl Book Tileset` para editá-lo em `Tilesets` no `Dashboard` ;
2. Selecione o *tile* no painel à esquerda conforme imagem:



Figura 3.9: Imagem de um tile selecionado em Tiled

3. Repare que, ao selecionar o(s) *tile(s)*, à direita aparecem vários *shapes* de colisões pré-configurados, bastando selecionar um deles. Clique na opção `Rect Right` para que o *shape* se ajuste à forma do *tile* selecionado, que ocupa mais a parte direita da área selecionada, conforme visualizado anteriormente;
4. Teste o jogo e veja como ficou o *tile* após a edição:

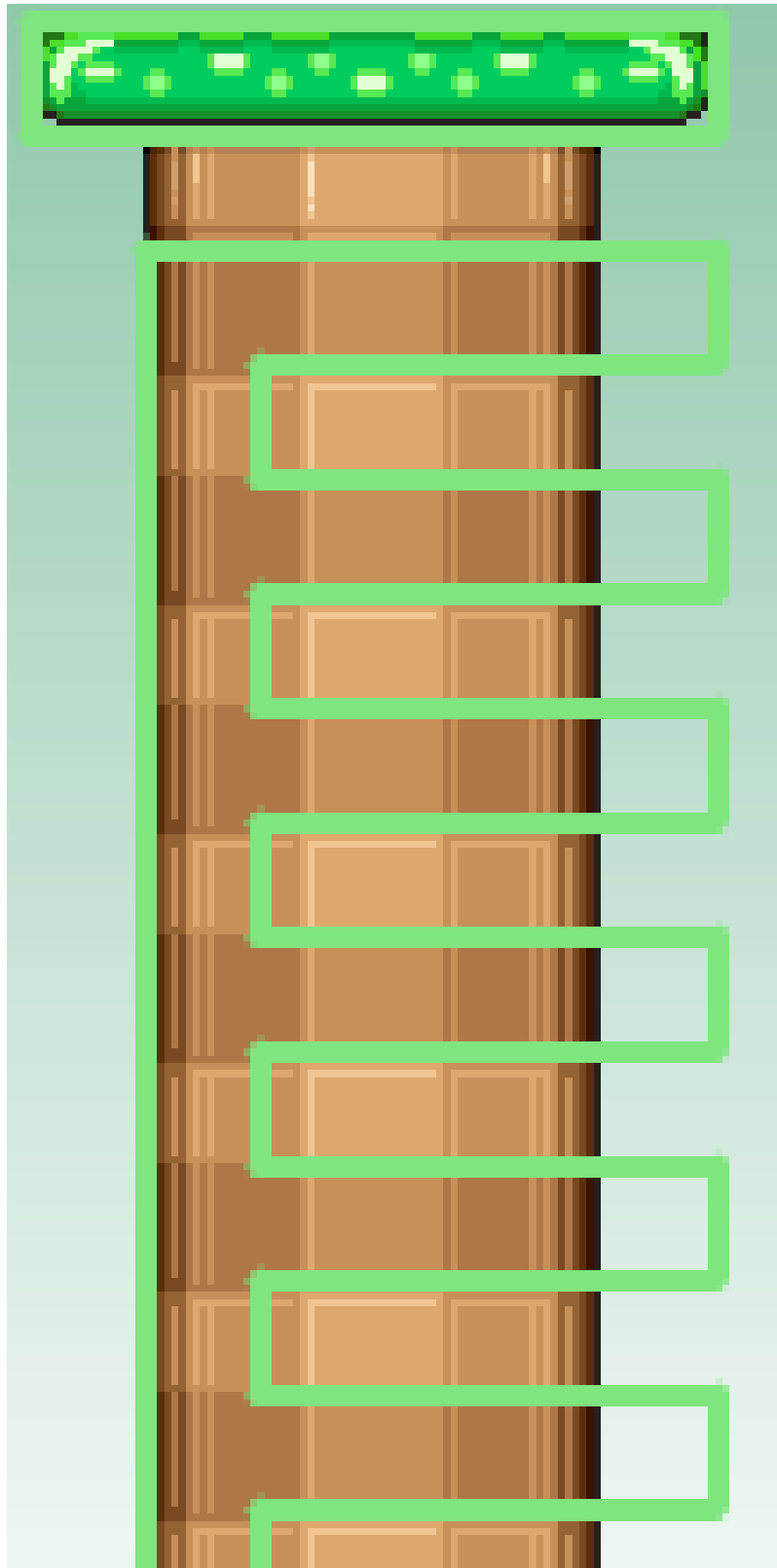


Figura 3.10: Imagem de um tile já editado no jogo

Faça você mesmo!

Agora é a sua vez. Realize o mesmo procedimento para os *tiles* do lado direito e selecione a edição adequada em `Collision Bounds`. Aproveite também para editar os outros *tiles* que foram incluídos na cena.

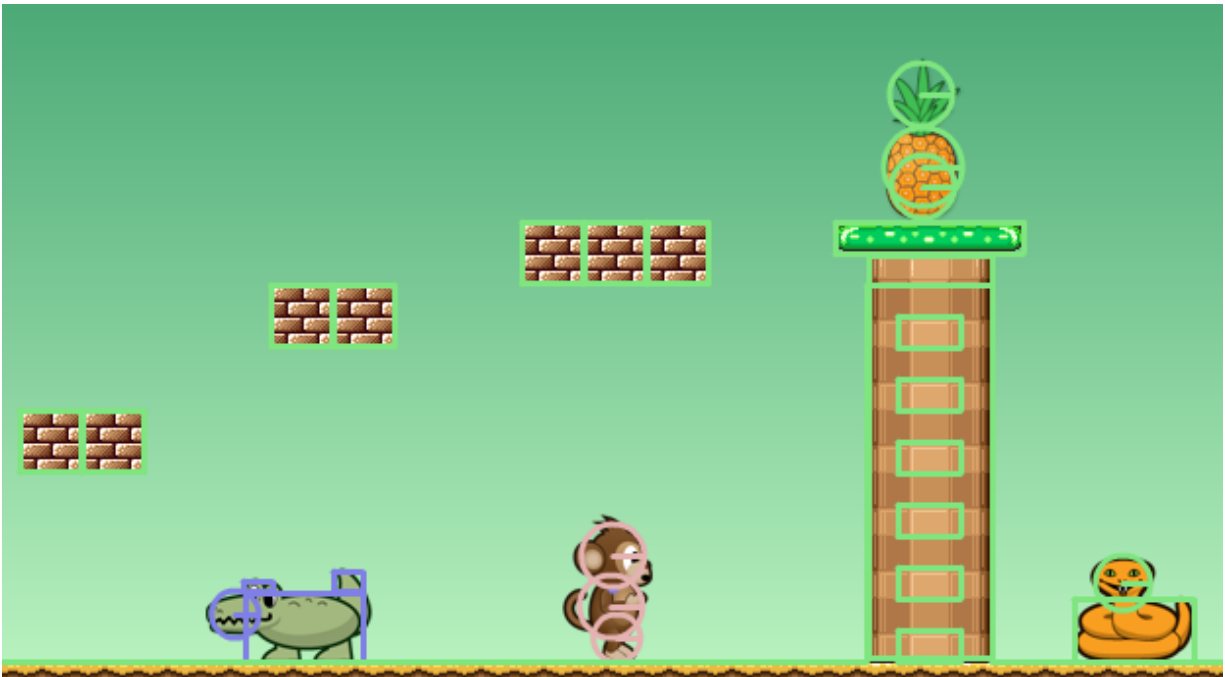


Figura 3.11: Tela do jogo com os tiles editados

3.3 Criando um grupo de colisão

Após a edição das colisões, é importante que agrupemos os *actors* e *tiles* para que, quando forem criadas as instruções para as colisões no jogo, não tenhamos que criá-las individualmente para cada um deles. Por exemplo: podemos criar um grupo de colisão para todos os vilões do jogo e, quando formos desenvolver as instruções, será necessário fazer apenas uma, para que nosso personagem `Stencyl Book Monkey` possa colidir com eles. Criar grupos

de colisões torna muito mais fácil a configuração posterior dos eventos de colisões.

Criaremos primeiramente um grupo de colisão para os vilões do jogo:

1. Na barra de ferramentas situada na parte superior da tela, clique em `Settings` ;
2. No painel à esquerda, clique na opção `Groups` ;
3. Na janela `Game Settings` , clique no botão verde `Create New` ;
4. No campo *Name*, digite **Inimigos** para nomear o grupo de vilões e confirme em `Create` . Repare que do lado direito da tela, no painel `Groups` , um novo grupo foi criado chamado `Inimigos` .



Figura 3.12: Lista de grupos de colisão do jogo

Faça você mesmo!

Agora é a sua vez. Crie um novo grupo de colisão chamado *Frutas*.

Configurando colisões para os grupos

Após criarmos os grupos, precisamos definir as colisões entre eles. O procedimento é muito fácil:

1. Na janela `Game Settings`, em `Groups`, selecione o grupo `Inimigos`;
2. Após ter selecionado o grupo `Inimigos`, o próximo painel é exibido:

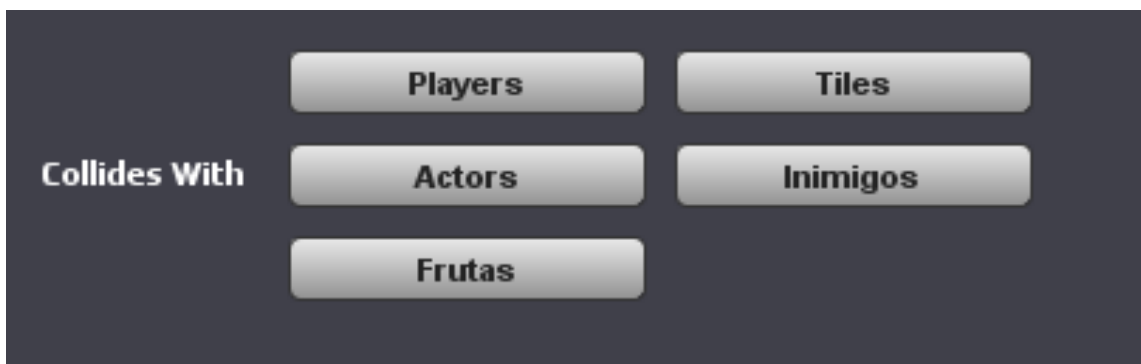


Figura 3.13: Tela de edição das colisões de grupos

3. Nesse painel, definiremos com quais grupos o grupo `Inimigos` vai colidir durante o jogo. Selecione as opções `Tiles`, `Actors` e `Frutas`. O grupo selecionado será exibido na cor verde.

Pronto! O grupo `Inimigos` colidirá apenas com os grupos previamente selecionados.

Faça você mesmo!

Agora é a sua vez! Defina as colisões para o grupo `Frutas`. Clique nas opções `Tiles`, `Actors` e `Inimigos`.

Inserindo os actors nos grupos de colisão

Estamos quase lá! Já criamos os grupos e definimos suas colisões, falta apenas incluir os *actors* nos seus respectivos grupos.

1. Abra o *actor* `Stencyl Book Croc` ;
2. Clique na guia `Properties` ; aqui nós visualizaremos as propriedades deste *actor*.
3. Em `Choose Group` , abaixo na tela, selecione o grupo `Inimigos` .

Faça você mesmo!

Pronto! Nosso jacaré foi incluído no grupo `Inimigos` . Fácil, não é mesmo? Inclua os *actors* `Stencyl Book Snake` também no grupo `Inimigos` e todas as frutas no grupo `Frutas` . Inclua o `Stencyl Book Monkey` no grupo `Actors` e defina sua colisão com os grupos `Inimigos` , `Frutas` e `Tiles` .

3.4 Utilizando e configurando sensores de colisão

A opção `Is a Sensor` , encontrada na guia `Collision` do *actor*, é utilizada para especificar que um determinado *actor* do jogo não sofre uma reação física quando colidir com outro *actor*. Por exemplo, quando um *actor* entrar em contato com um item colecionável, como as frutas no nosso jogo, elas não vão impedir seu deslocamento, então ele vai se movimentar normalmente.

Habilitaremos o sensor de colisão para nossas frutas. Posteriormente, para simular que o macaco as coletou, elas desaparecerão no jogo quando nosso *actor* `Stencyl Book Monkey` entrar em contato com elas. Iniciaremos a configuração para o *actor* `Stencyl Book Banana` :

1. Clique duas vezes em `Stencyl Book Banana` localizado em `Actor Types` do `Dashboard` ;
2. Clique em `Collision` na parte superior da tela;
3. Selecione o círculo mais à esquerda, conforme imagem a seguir:

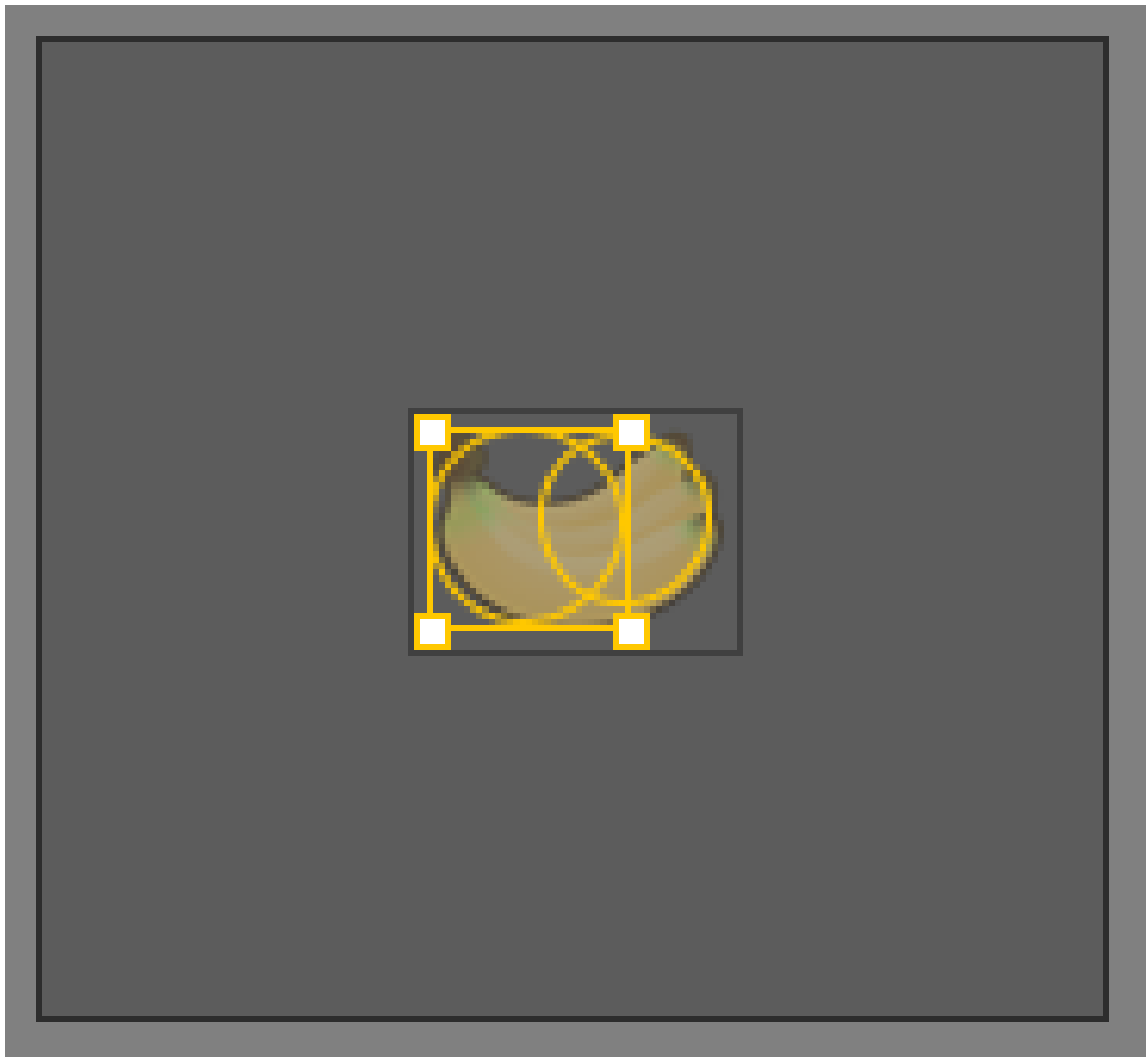


Figura 3.14: Visualização dos shapes de colisão

4. Em `Physical Properties` , à direita da tela, habilite `Is a Sensor?` ;
5. Faça o mesmo procedimento para o círculo da direita;
6. Salve e teste o jogo.

Pronto! Assim que o nosso macaquinho entrar em contato com a banana, ela não impedirá sua passagem.

Faça você mesmo!

Habilite a opção `Is a Sensor` para as outras frutas. Não se esqueça de habilitá-la para cada *shape* de colisão.

Neste capítulo:

- aprendemos a editar *shapes* de colisão;
- a criar grupos de colisões;
- a utilizar os sensores de colisão.

No próximo capítulo, nós nos aprofundaremos na compreensão e desenvolvimento de eventos e *behaviors*.

CAPÍTULO 4

Criando behaviors

Neste capítulo, vamos desenvolver nossos próprios *behaviors*. No capítulo 2, utilizamos *behaviors* pré-configurados pelo software para fazer com que nosso macaquinho se movimentasse e também para que ele não saísse de cena. Dessa vez, vamos ampliar nossa compreensão sobre como utilizar e configurar os blocos de instruções para os *behaviors*.

Para relembrar, *behaviors* são respostas a eventos que acontecem durante o jogo e que proporcionam que sua mecânica funcione. Por exemplo: quando um *actor* colidir com o outro, o que acontecerá? Ele vai desaparecer? Quando ele vai desaparecer? Como? Como cada *actor* se comportará? As respostas a essas perguntas serão respondidas neste capítulo. É de fundamental importância a compreensão e utilização correta dos eventos e *behaviors* para que o jogo funcione corretamente, para que o *gameplay* fique conforme o projeto proposto, para que os eventos e suas repostas aconteçam adequadamente segundo as instruções determinadas.

4.1 Desenvolvendo uma instrução de colisão

Como vimos anteriormente, existem *behaviors* pré-configurados para movimentos e saltos, mas em nosso jogo gostaríamos de criar mais possibilidades para dinamizá-lo. Queremos que nosso macaco colida com outros *actors* e colete itens na cena, por exemplo. Como

estes comportamentos não vêm pré-configurados, é possível criar nossos próprios *behaviors*. É o que vamos fazer agora.

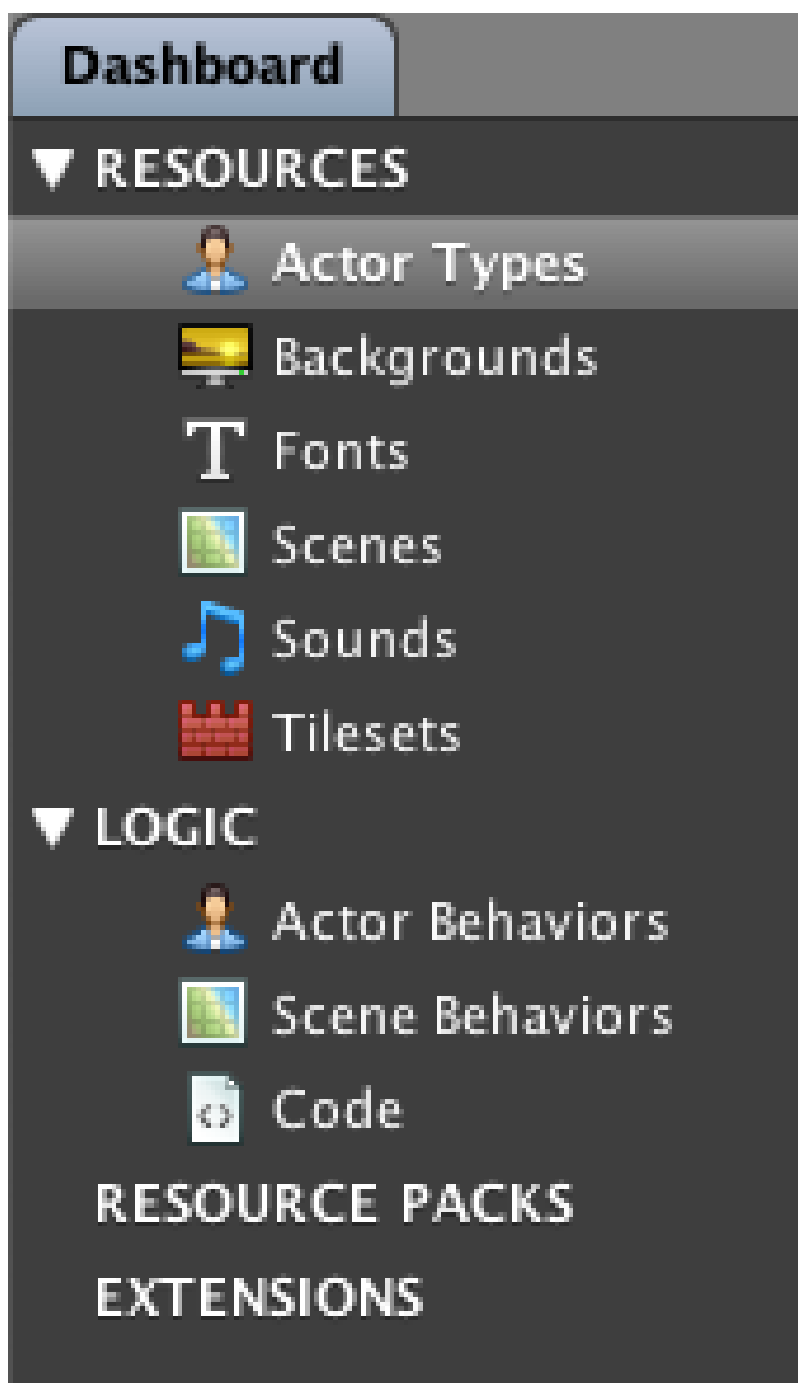


Figura 4.1: Painel Dashboard

Na parte inferior do painel `Dashboard` , na seção `Logic` , temos dois tipos de *behaviors*: *behaviors* direcionados aos *actors* e às cenas. Nessa primeira etapa, criaremos um primeiro *behavior* para o *actor* `Stencyl Book Monkey` para que ele possa coletar as frutas durante o jogo. Vamos ver as etapas:

1. Clique em `Actor Behaviors` . Você deve ter visualizado que já existem alguns *actors behaviors* criados, pois utilizamos no capítulo 2 alguns já pré-configurados.
2. Clique no botão verde à direita na parte superior da tela chamado `Create New...` . A tela seguinte é dividida em 3 áreas:

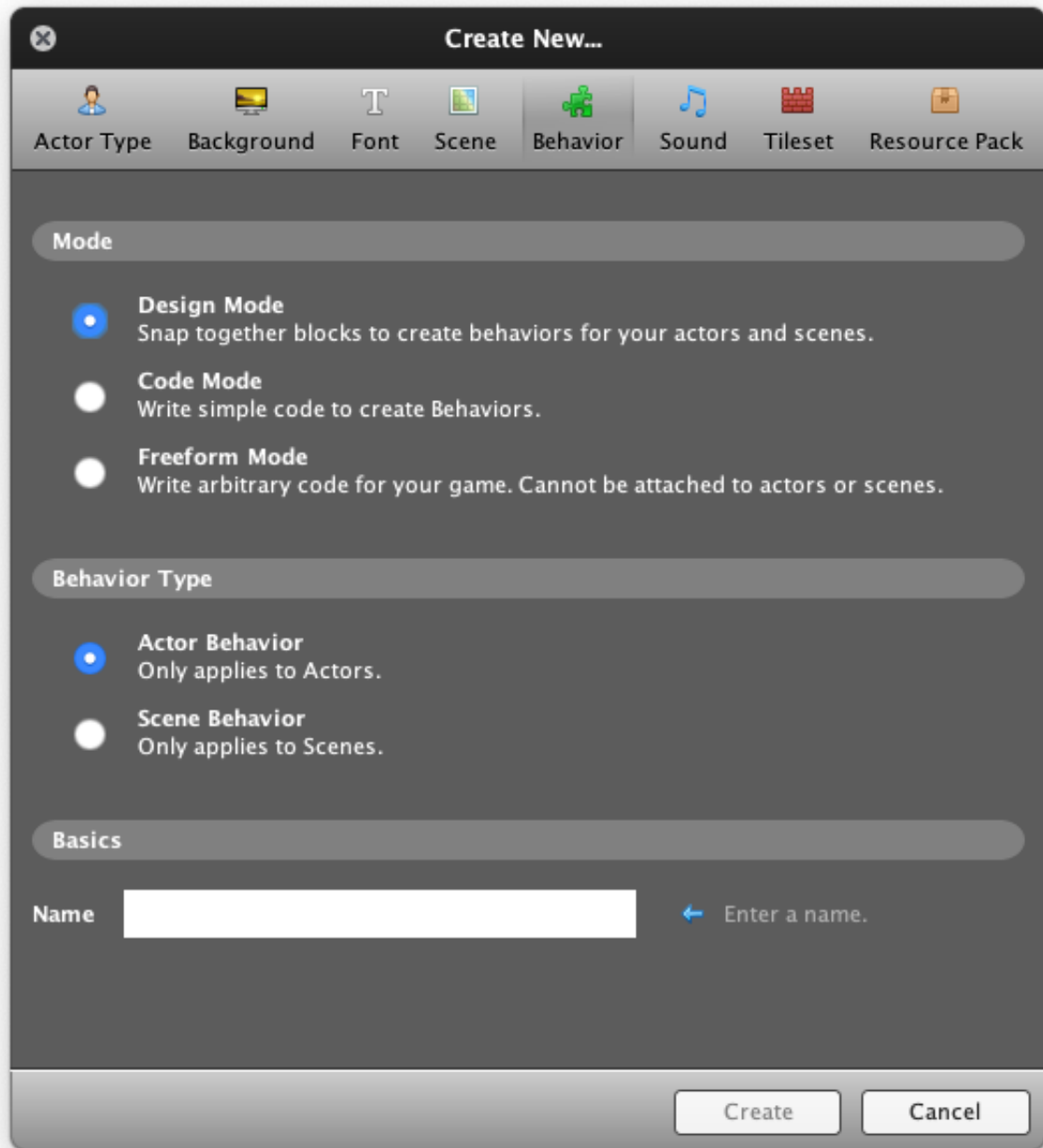


Figura 4.2: Tela de criação de um novo actor behavior

- **Mode:** utilizaremos aqui o modo *Design Mode* (modo design) onde as instruções são desenvolvidas por meio dos blocos de instrução.
- **Behavior Type:** definição sobre o tipo de *behavior*
- **Basics:** inclusão do nome do *behavior*

3. No campo `Name` , digite *Colisoes* e clique em `Create` ; Evitaremos a utilização de acentuação para evitar erros de interpretação da linguagem e também porque o próprio *Stencyl*, em muitos casos, impede sua utilização. Utilizaremos esse *actor behavior* para desenvolvermos nossos *behaviors* relacionados às colisões do nosso *actor* *Stencyl Book Monkey* .
4. A tela a seguir será mostrada:



Figura 4.3: Interface para criação de eventos e behaviors

A tela é dividida em 3 partes: à esquerda, escolhemos o evento a ser criado; na parte do meio, inserimos os blocos de instrução; e à direita, os selecionamos. Vamos com a próxima etapa:

5. Selecione `+Add Event` no canto superior esquerdo da tela para criar um novo evento. Neste menu selecionaremos nossos eventos. Veja que eles são divididos por categorias:
 - *Basics*: os principais eventos do software. Muitos deles se iniciam por aqui.

- *Input*: eventos para a utilização de dispositivos como teclados ou mobiles.
 - *Time*: eventos para o controle do tempo.
 - *Collisions*: eventos relacionados às colisões.
 - *Sound*: eventos relacionados ao som inserido no jogo.
 - *Mobile*: eventos relacionados à monetização do jogo.
 - *Attributes*: eventos para criação de atributos para as instruções
 - *Advanced*: grupo para a configuração avançada e personalizada de eventos.
6. Para que nosso macaquinho colida com as frutas, utilizaremos o evento o grupo *Collisions* . As frutas estão dentro de um grupo de colisão, então, selecionaremos a opção *Member of Group* . Veja:

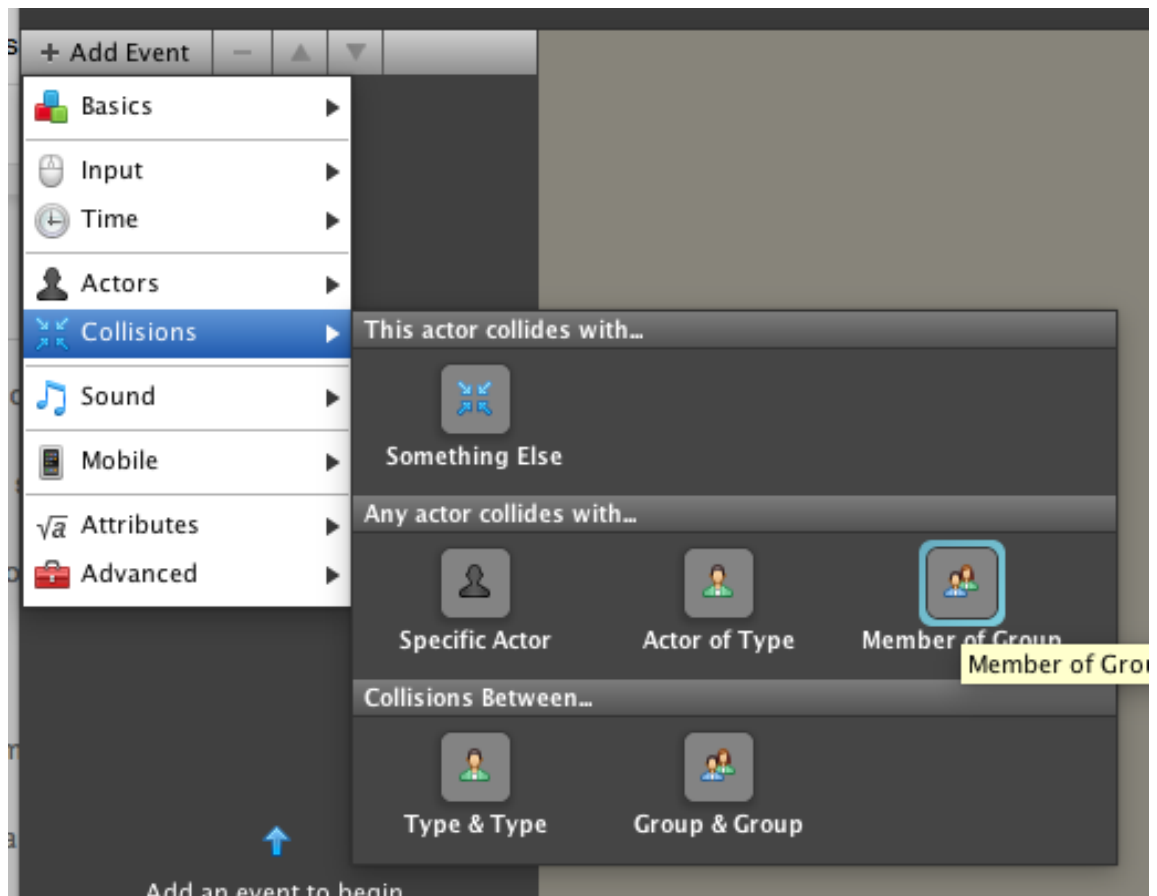


Figura 4.4: Eventos da categoria colisões

7. O evento foi criado. Vamos renomeá-lo? Clique duas vezes em `Actor - Group` e renomeie o evento para `Colidir Frutas`. O nome é apenas um rótulo, então você pode acentuar normalmente;
8. No bloco de instrução laranja, no meio da tela no campo `when` (quando), selecione `self`. A instrução `when` refere-se ao momento em que a colisão vai acontecer com um membro do grupo `Frutas`. *Self* é porque essa instrução funcionará anexada a um determinado *actor*, nesse caso, o `Stencyl Book Monkey`.
9. Em `Actor Group`, selecione `Choose Group`, clique em `Frutas` e dê `ok`. Selecionamos aqui o grupo que sofrerá a colisão.
10. Salve o jogo.

Até esse momento, criamos o evento para que sempre que um *actor* atingir um item do grupo `Frutas`, algo aconteça. Vamos agora configurar uma ação de resposta:

1. Na parte inferior da tela, do lado direito, clique em `Pallette` para visualizar os blocos de instrução;
2. A janela a seguir apresenta as categorias dos diversos blocos de instrução (repare que antes do nome da categoria visualizamos também a cor dos blocos para facilitar sua localização). Acima dos blocos também visualizamos uma linha de pesquisa. Vamos clicar em `Actor` conforme a próxima imagem para que possamos acessar os blocos de instrução dessa categoria;



Figura 4.5: Categorias dos blocos de instrução

3. A resposta ao evento de colisão do `Stencyl Book Monkey` com as frutas é fazer as frutas desaparecerem. Vamos localizar o bloco adequado a esta instrução. Selecione a opção `Properties` logo abaixo das categorias;
4. Para configurar nosso evento, precisaremos mover os blocos de instrução para a tela principal. Para movê-los, basta clicar e arrastar o bloco desejado. Arraste o bloco `kill Actor` para o bloco laranja. Essa instrução serve para eliminar o *actor* do jogo. Veja como ficou:

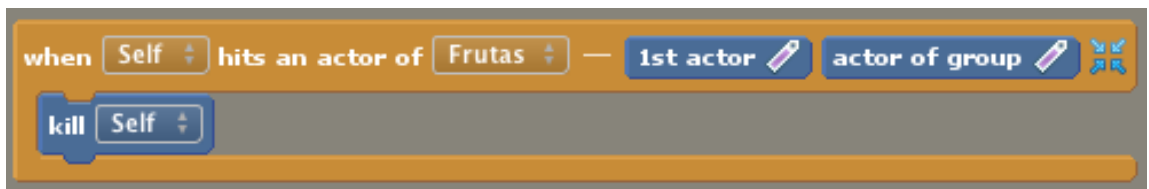


Figura 4.6: Instrução para eliminar actors da cena

5. Arraste o bloco `actor of group` do próprio bloco laranja visto acima e solte dentro da caixa `self` da opção `kill`, conforme imagem:



Figura 4.7: Instrução para eliminar actors da cena

Vamos compreender o que fizemos? Sempre que nosso *actor* Stencyl Book Monkey colidir com um *actor* do grupo Frutas, o membro deste grupo desaparecerá.

6. Para finalizar, vamos anexar essa instrução ao Stencyl Book Monkey, pois é ele quem vai coletar as frutas. Clique no botão verde à direita da tela chamado Attach to Actor Type;



Figura 4.8: Botão para anexar o evento ao actor

7. Selecione nosso herói e confirme.

8. Salve e teste o jogo.

Caminhe com nosso herói e perceba que agora as frutas somem quando em contato com ele.

4.2 Criando um evento de colisão entre herói e inimigo

Viu como é fácil criar um *behavior* para nosso macaquinho coletar frutas? Nosso herói já pode coletar as frutas ou qualquer outro item que você inserir no jogo, bastando que os *actors* estejam em seus grupos corretos e com os *shapes* de colisão editados

adequadamente. Dessa vez, quem vai desaparecer é o nosso herói quando colidir com algum actor do grupo inimigo. O procedimento é semelhante ao aprendido anteriormente. Vamos utilizar o mesmo *actor behavior* Colisoes . Vamos lá?

1. Para acessar novamente o *actor behavior* Colisoes , clique Actor Behaviors NO Dashboard ;
2. Abra o evento Colisões clicando duas vezes sobre ele;
3. Vamos acrescentar um novo evento. Clique em +Add Event no canto superior esquerdo da tela, desça o mouse até Collisions e selecione novamente Member of Group ;

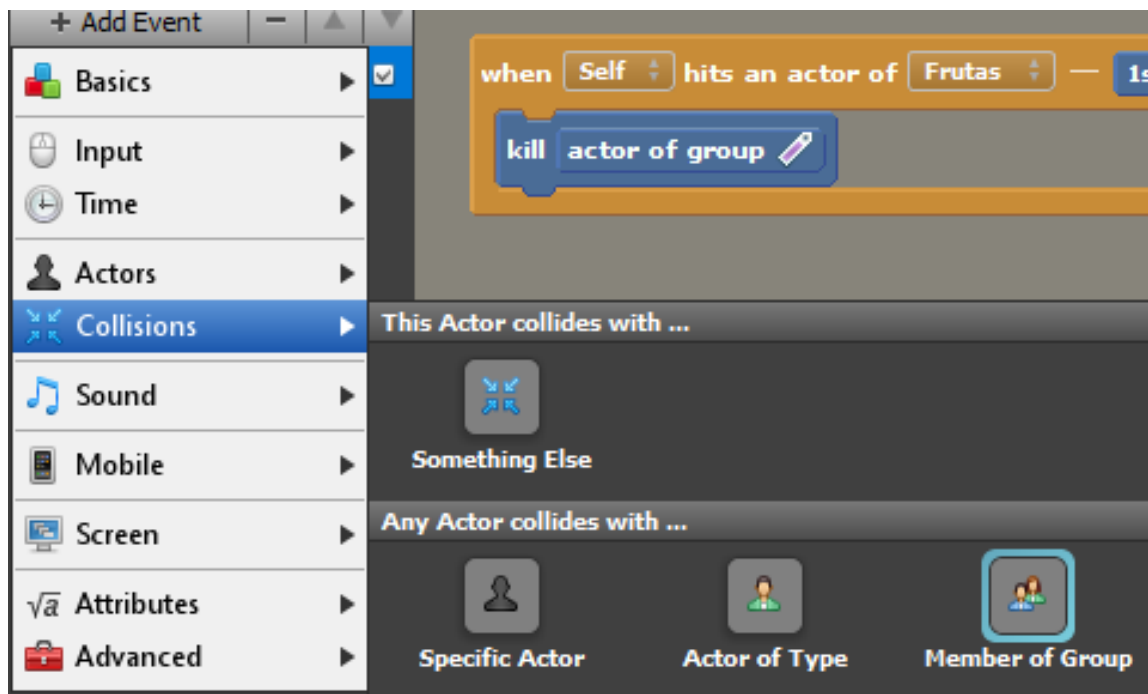


Figura 4.9: Eventos da categoria colisões

4. Renomeie-o para Colidir Inimigos ;
5. No bloco laranja, em Actor Groups , selecione o grupo Inimigos ;
6. Clique em OK ;
7. Clique em Pallette no canto inferior direito da tela;
8. No botão Actor , na guia Properties , arraste o bloco kill actor para o bloco laranja. Veja:



Figura 4.10: Eventos da categoria colisões

9. Vamos aprender a renomear este *actor behavior*? Na guia *Properties*, acima do bloco laranja, mude o nome da propriedade do *behavior* para *Colisoes do Ator*;
10. Clique em *Apply Changes*;
11. Salve e teste o jogo.

Pronto! Toda vez que nosso personagem *Stencyl Book Monkey* colidir com um inimigo, dessa vez, quem sairá de cena é ele.

4.3 Inserindo novos actors na cena

Nosso *actor* já sabe coletar itens e também, se não tomar cuidado, pode desaparecer do jogo. E agora que estamos compreendendo o método de construção de *behaviors* por blocos e também a desenvolvê-los, vamos aumentar os obstáculos para nosso herói. Dessa vez vamos adicionar novos *actors* para atrapalhar sua aventura e aumentar seus desafios. Baixe o actor *Stencyl Book Statue* lá no *StencylForge* e, depois, siga as etapas a seguir:

1. Vamos criar um novo grupo de colisão chamado *Estátuas* e configurá-lo para colidir com todos os outros grupos (consulte a seção 3.3 *Criando um grupo de colisão*);
2. Nas propriedades do *actor Stencyl Book Statue* adicione-o ao grupo criado;
3. Dessa vez, vamos criar um *behavior* para a cena, pois é lá que os novos *actors* aparecerão. Então, no *Dashboard*, vamos clicar em *Scene Behaviors*;

4. Como ainda não temos nenhum *behavior* criado para cenas, clique em *This game contains Logic. Click here to create one* OU em *Create New* acima e à direita;
5. Vamos nomear esse *Scene Behavior* como *Obstaculos Estatuas* e clicar em *Create* ;
6. Crie um novo evento conforme a imagem a seguir. O evento *Every N seconds* , ou seja, a cada período de segundos, nos permitirá definir um determinado tempo para o aparecimento das estátuas:

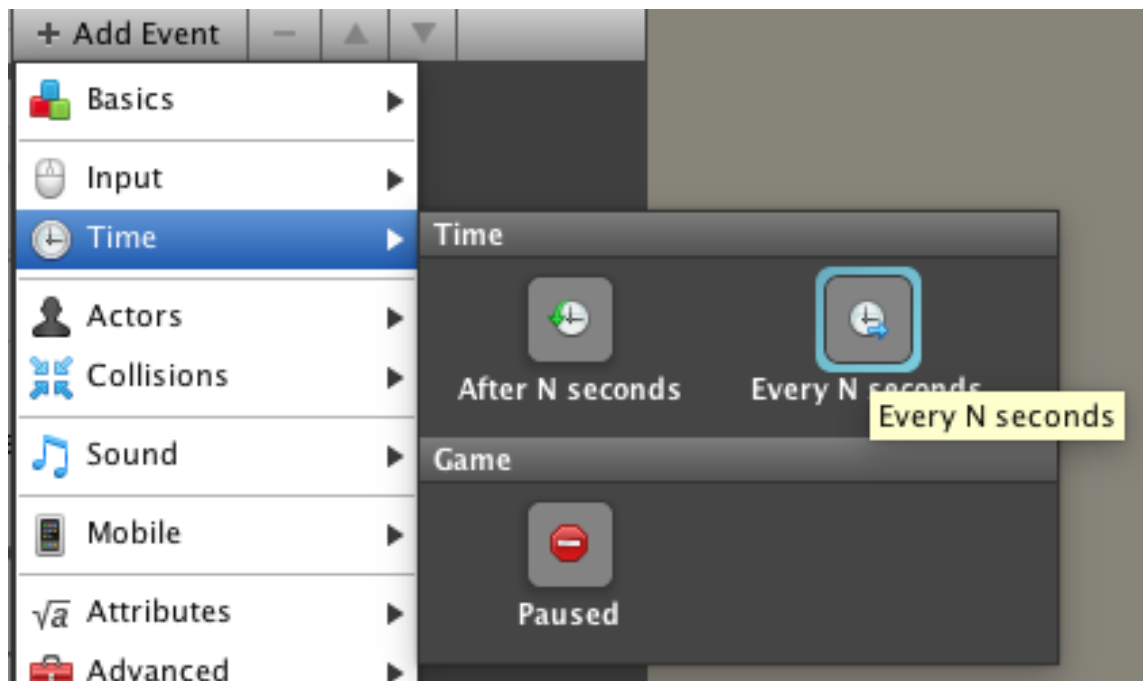


Figura 4.11: Eventos da categoria Time

7. Mude o nome de *Every N Seconds* para *Obstaculos* ;
8. Vamos especificar um tempo para o aparecimento das estátuas. Digite 3 (segundos) no campo editável do bloco laranja;
9. Vamos clicar no botão *Scenes* à direita e, na guia *Actors* , arraste para o bloco laranja o bloco vermelho *create Actor Type at... Front* , conforme abaixo. Nesse bloco especificaremos qual *actor* aparecerá na cena:



Figura 4.12: Bloco de criação de actors na cena

10. Em `Actor Type` , vamos selecionar o actor que aparecerá na cena. Clique em `Stencyl Book Statue` ;
11. Preencha as coordenadas x e y com as posições onde o *actor* aparecerá na cena. A coordenada x refere-se à distância do início da cena na horizontal, e a coordenada y sua distância na vertical. O ponto zero das duas coordenadas iniciam-se no canto superior esquerdo da cena. Com os valores a seguir, as estátuas surgirão a cada 03 segundos a 320 px de distância do ponto zero na horizontal e a 10 px na vertical:

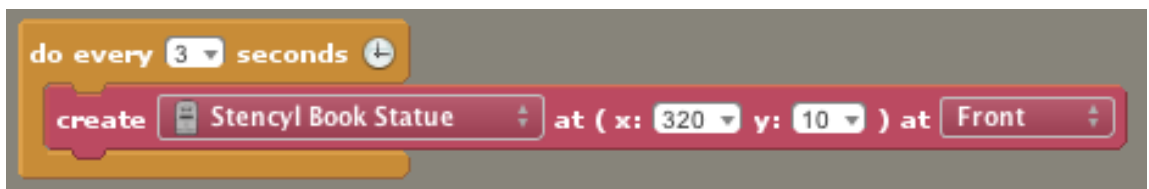


Figura 4.13: Bloco com as coordenadas para o aparecimento do actor inseridas

12. Anexe o evento à `Cena01` clicando em `Attach to Scene` à direita da tela;
13. Salve e teste o jogo.

Repare que, a cada 3 segundos, as estátuas surgem na cena `Cena01` . Mas, notou que nós temos um pequeno probleminha? As pedras estão caindo sempre no mesmo lugar da cena, devido às coordenadas que inserimos. Vamos ajustar para caírem conforme nosso herói caminha no jogo?

4.4 Ajustando os actors ao deslocamento da câmera

Como você deve ter notado, conforme nosso *actor* se desloca pela cena, as pedras ficam para trás, pois definimos um local fixo para elas caírem. Vamos fazê-las agora seguirem as coordenadas da câmera, e não mais da cena:

1. Continuando no evento `obstaculos`, vamos clicar na seta de seleção da coordenada `x`, clicar em `Math` para selecionar o bloco verde de soma, conforme imagem a seguir. *Math* são instruções envolvendo cálculos matemáticos. Utilizaremos bastante conforme caminharmos no livro:

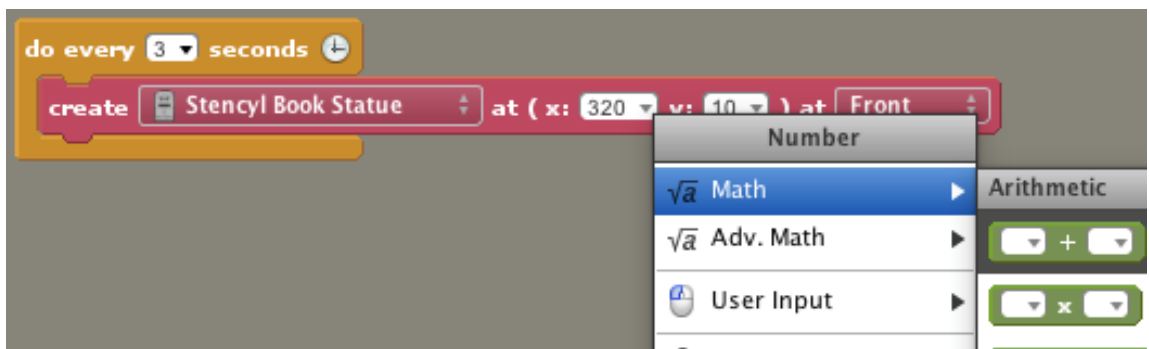


Figura 4.14: Seleção da instrução aritmética de soma

2. Na coordenada `x` do bloco verde, vamos descer até `Scene` e selecione `x of camera`. Agora a coordenada `x` será do ponto 0 da câmera;
3. Digite o valor "320" no outro bloco à direita para especificar a distância do 0 da coordenada `x` da câmera.
4. Salve e teste o jogo.

Agora as coordenadas acompanham horizontalmente o `x` da câmera, conforme nosso personagem caminha na cena. Experimente outros valores nas coordenadas!

4.5 Inserindo actors em coordenadas aleatórias

Nosso jogo está ficando cada vez mais interessante, não é mesmo? Após criarmos *behaviors* para o aparecimento de novos *actors* e também ajustá-los ao deslocamento da câmera, vamos fazê-los surgir em posições diferentes, de forma aleatória, para que parem de cair na mesma posição da câmera:

1. Continuando na mesma instrução do tópico anterior, vamos clicar na caixa editável do bloco verde, onde está o valor 320, e inserir o bloco `random integer between`, conforme imagem. Esse bloco permitirá inserir intervalos para o aparecimento aleatório das estátuas:

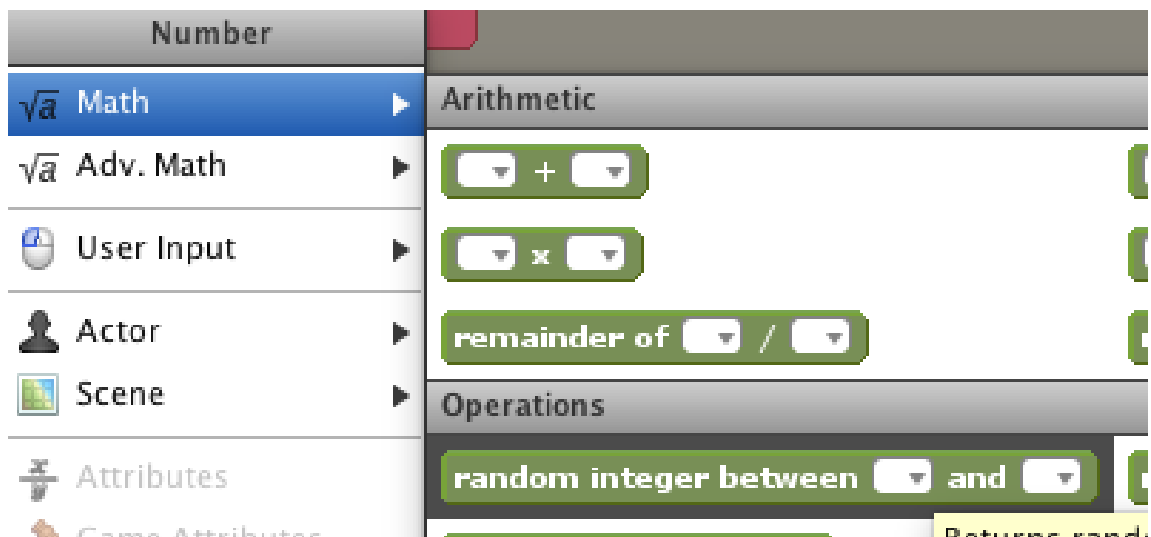


Figura 4.15: Seleção de bloco de aleatoriedade

2. Digite o número 0 na primeira caixa após a palavra *between* e na outra caixa à direita clique na seta de seleção, desça até `Scene` e selecione `screen width` (largura da tela), ou seja, os novos *actors* agora surgirão entre a coordenada 0 da câmera e a sua largura.
3. Teste o jogo.

Repare que as estátuas surgem de maneira aleatória e sempre no intervalo entre o início e o fim da tela.

4.6 Eliminando as estátuas da cena

Estamos chegando ao final deste capítulo. Fizemos novos *actors* aparecerem de forma aleatória mas eles continuam na cena. Nas próximas etapas vamos, após um determinado tempo, fazê-los desaparecer:

1. Vamos continuar no evento `obstaculos`. Selecciona a categoria de instruções `Flow` na `Pallette`. Nesta categoria encontraremos várias instruções relacionada ao fluxo do jogo, como fórmulas condicionais ou relacionadas a intervalos de tempo;
2. Um pouco mais abaixo, clique na guia `Time` e localize o bloco `do after seconds`;
3. Arraste para dentro do bloco laranja e abaixo do bloco vermelho. Veja a imagem:

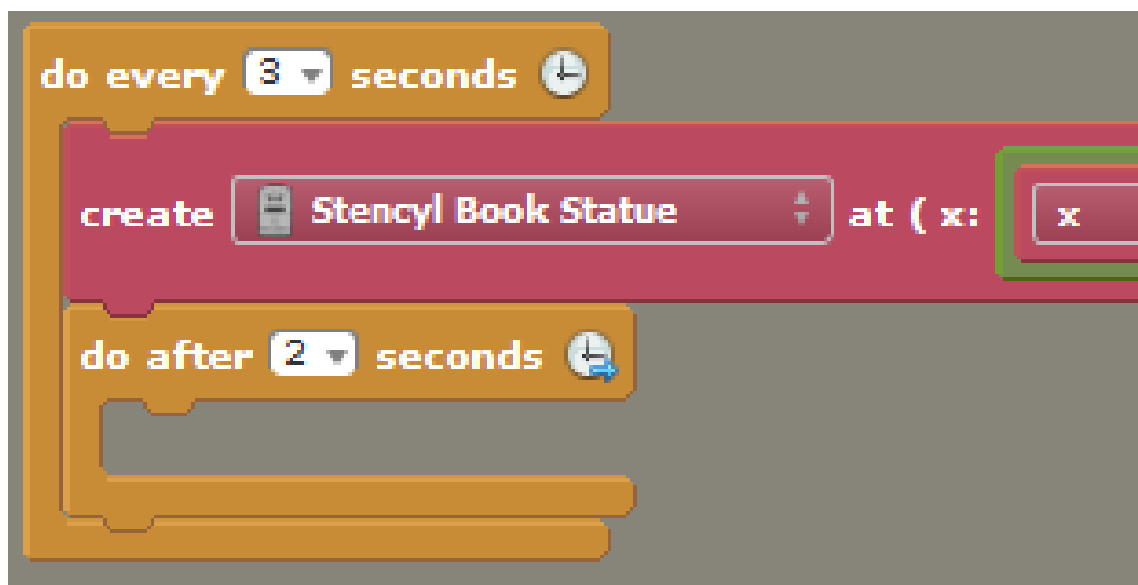


Figura 4.16: Instrução relacionada a intervalos de tempo

4. A instrução, até o momento, demonstra que algo acontecerá ao *actor* criado após um intervalo de segundos. Vamos inserir no campo editável *seconds* o valor 2 e arrastar o bloco *kill* da categoria *Actors*, lembra?

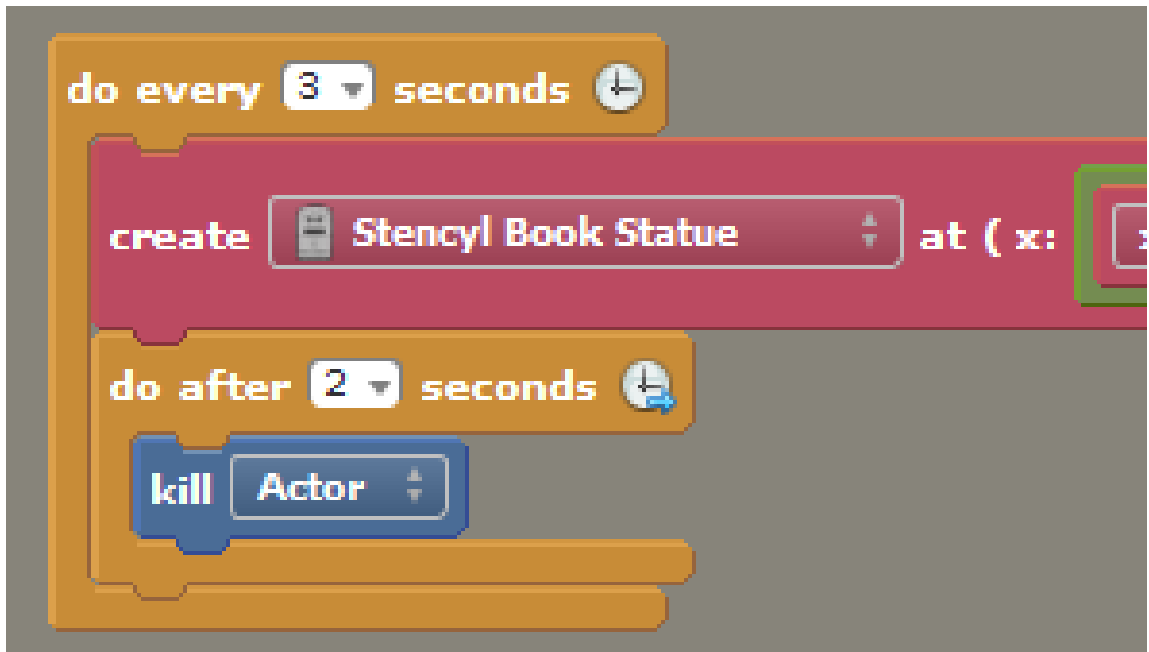


Figura 4.17: Instrução como intervalo de tempo

5. Para concluir, vamos especificar na opção *actor* do bloco *kill* que o *actor* que eliminado será o último criado na cena. Selecione a opção *Last Created Actor*.
6. Salve e teste o jogo.

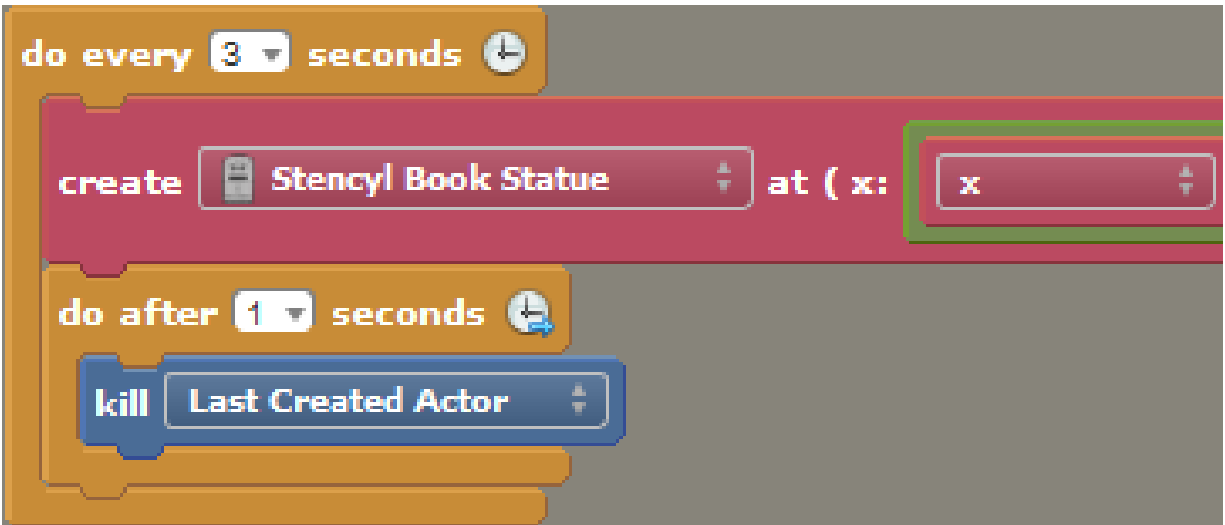


Figura 4.18: Instrução para eliminação do actor da cena

Perceba que dois segundos após aparecerem as estátuas somem.

Desaparecendo lentamente

Vamos inserir um efeito para os *actors* desaparecerem mais lentamente da cena? Basta inserir o bloco `fade over` antes de eliminá-los de vez. Veja o procedimento:

1. Ainda no evento `Obstaculos`, mantenha pressionada a tecla `alt` (Windows) ou `option` (Mac OS), tire uma cópia do bloco `do after seconds` e posicione-o abaixo. Veja o exemplo:

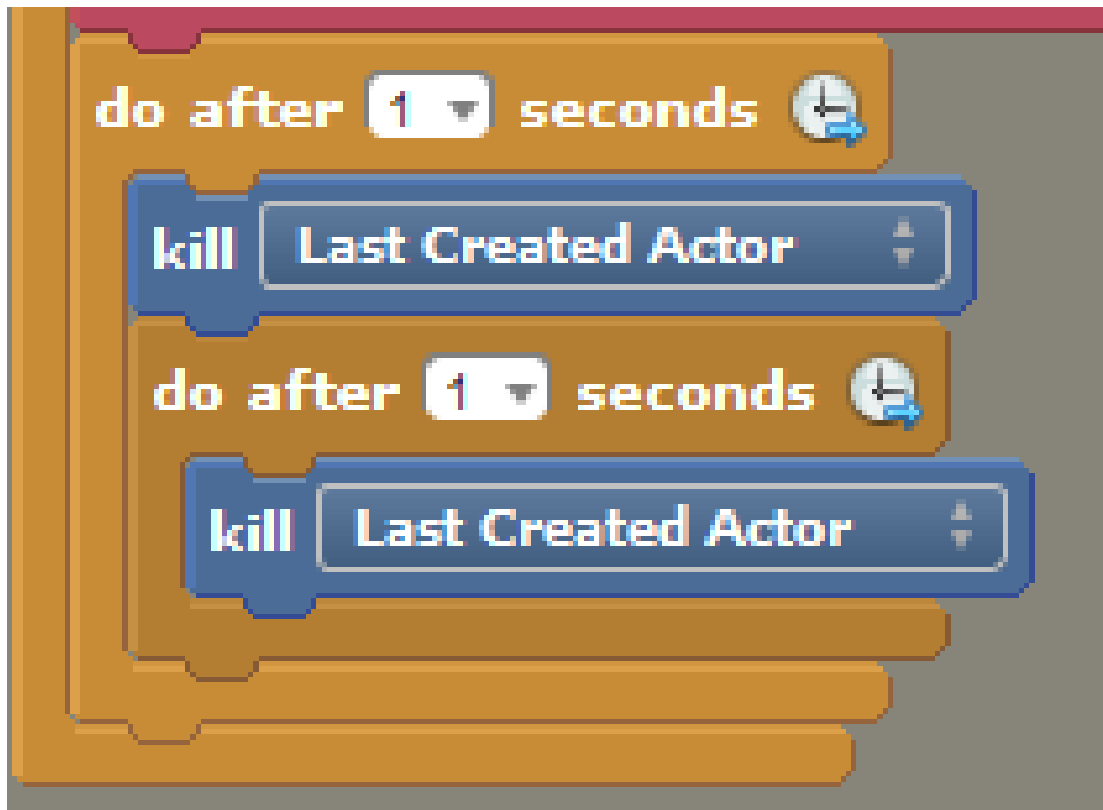


Figura 4.19: Bloco copiado e posicionado

2. Agora vamos substituir o bloco `kill` de cima pelo bloco `fade`. Clique e arraste o bloco `kill` para fora do bloco laranja. O bloco conectado abaixo também será movido:

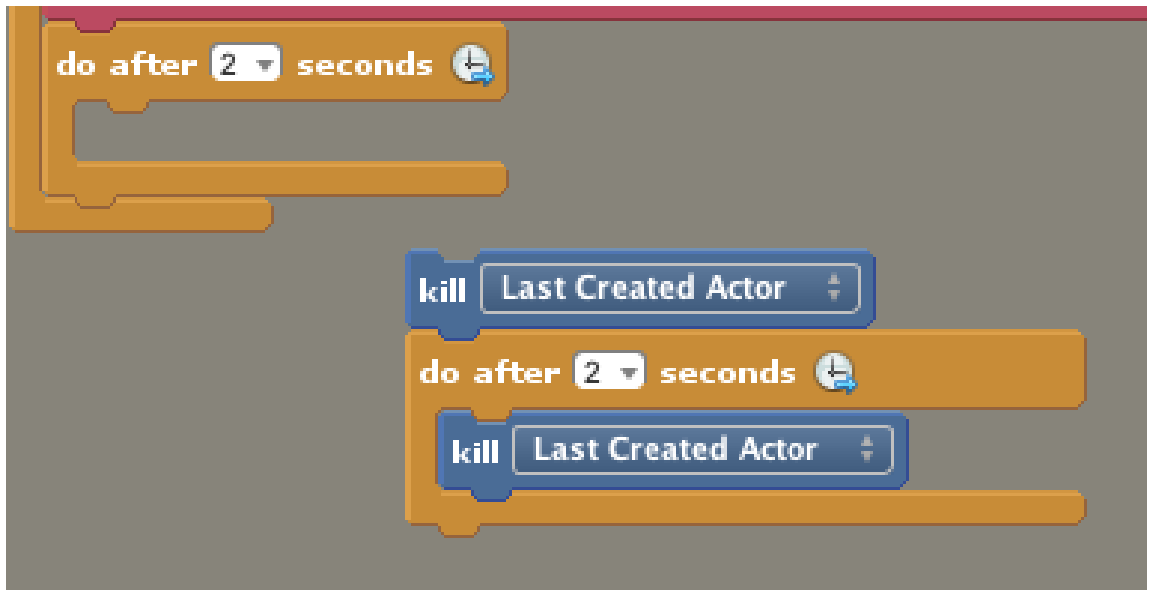


Figura 4.20: Bloco movido da instrução

3. Desconecte o bloco `do after seconds` do bloco `kill` movido;
4. Na *Pallette*, à direita, vamos clicar na categoria `Actors` e na guia `Tweening`. Nessa guia encontramos efeitos para os *actors*;
5. Mova o bloco `fade in actor over sec using` e conecte-o sobre o bloco `do after`;
6. Vamos configurar para que as estátuas (`Last Created Actor`) desapareçam (`fade out`) entre 01 segundo (`over sec`). Veja:

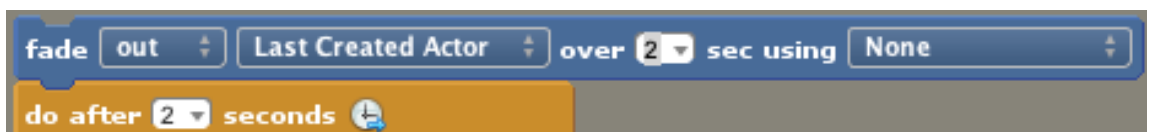


Figura 4.21: Configuração do bloco fade

7. Vamos levar este bloco inteiro de volta ao outro bloco `do after`. Confira como ficou e teste o jogo.

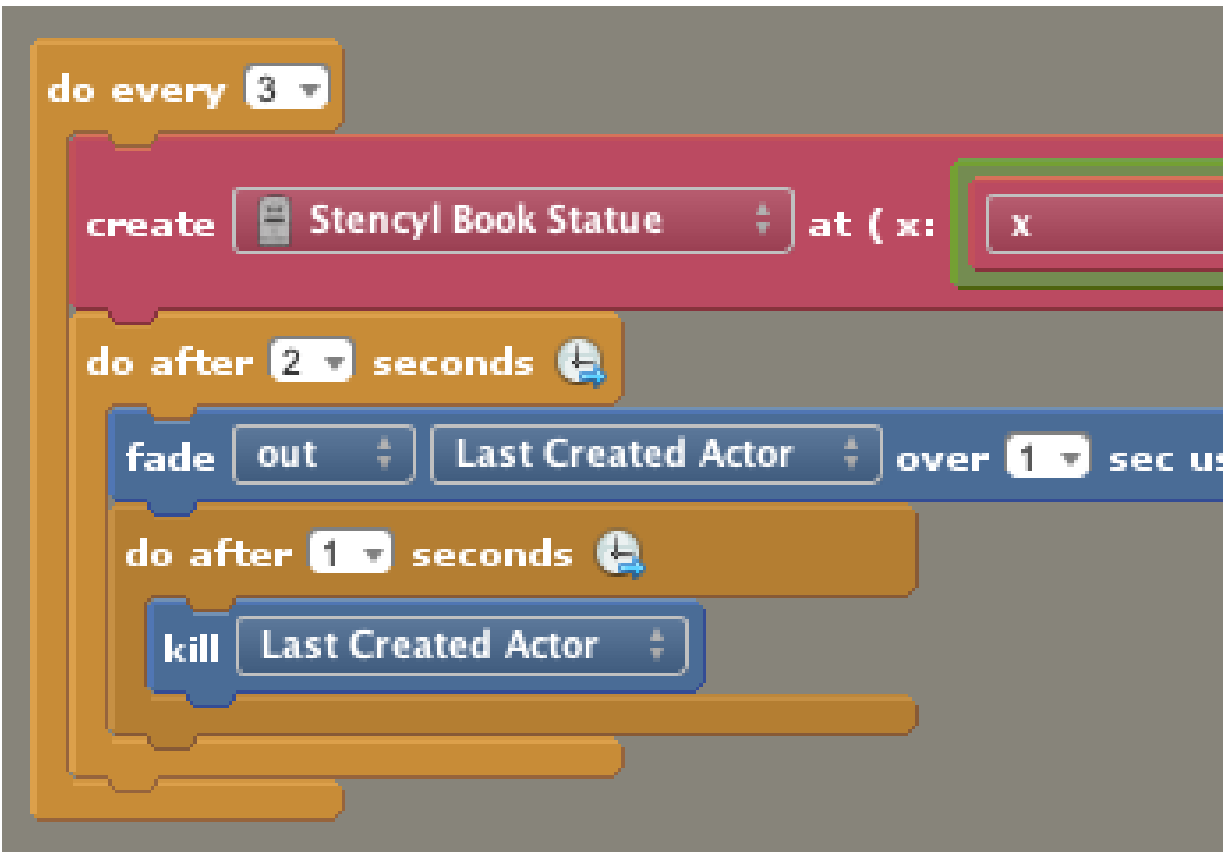


Figura 4.22: Instrução para eliminação do actor com efeito

Pronto! Concluímos a instrução. Estátuas surgirão a cada 3 segundos no caminho do nosso macaco. Após 2 segundos elas começam a desaparecer em um intervalo de 1 segundo e são eliminadas após 1 segundo. Experimente outros valores!

Faça você mesmo!

Agora é a sua vez. Aplique o efeito *fade out* quando as frutas forem coletadas, seguindo as etapas anteriores. Veja o exemplo a seguir. A opção *Last Collided Actor* fará com que desapareça o último *actor* que colidir com nosso herói.



Figura 4.23: Eliminação dos actors Frutas com efeito

Caso queira, crie também um novo evento chamado `Colidir Estátuas` no *actor behavior* `Colisoes` para que as estátuas, ao colidirem com o nosso macaquinho, elimine-o da cena.

Neste capítulo:

Aprendemos a criar eventos com os blocos de instrução para:

- colisões;
- inserção e eliminação de novos *actors* na cena;
- inserção de efeitos nas instruções.

No próximo capítulo vamos aprender a controlar o tempo no jogo. Até lá!

CAPÍTULO 5

Cronometrando o tempo no jogo

Neste capítulo, vamos trabalhar com eventos relacionados à contagem do tempo no jogo. Cronometrar o tempo com uma contagem regressiva aumenta o desafio do nosso herói para completar o nível, pois além de desviar dos seus inimigos terá que coletar as frutas dentro de um prazo determinado.

Vamos iniciar inserindo no jogo um evento para que se inicie uma contagem regressiva no primeiro nível. O tempo será contado de forma decrescente para que nosso jogador veja quanto tempo ainda possui para realizar seu objetivo, se não será eliminado do jogo. Sem a contagem, fica muito fácil coletar as frutas, não é verdade?

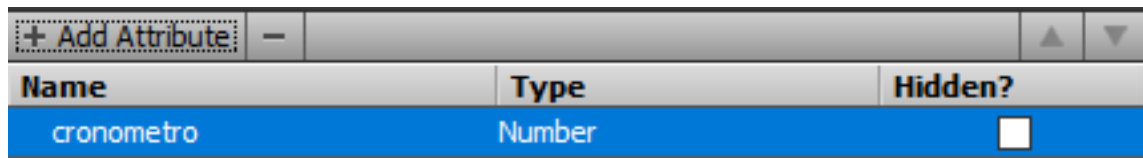
5.1 Desenvolvendo uma instrução de tempo

Para desenvolvermos uma instrução para a contagem do tempo, utilizaremos um novo evento chamado `Time`. Esse evento contará o tempo para a cena, portanto, precisaremos criar um *Scene Behavior*. A partir deste capítulo, vamos diminuir o detalhamento das etapas pois você já está mais familiarizado com o software por meio das atividades anteriores. Vamos lá:

1. Vamos criar um novo *Scene Behavior* chamado `Contagem Regressiva` ;
2. Vamos primeiramente criar um atributo com um valor para utilizarmos na contagem. Um atributo é uma propriedade, uma variável que possui um valor que, vinculado a um objeto, pode sofrer alterações. No nosso caso, incluiremos nesse atributo um valor para a contagem do tempo, que pode ser alterado em qualquer momento. Nas categorias de instruções, à direita da

tela, clique em `Attributes` e em seguida em `Create an Attribute...` ;

3. Na janela `Create An Attribute...` , digite o nome **cronometro** e abaixo selecione o tipo `Number` . Incluiremos depois um valor numérico neste atributo relacionado à contagem do tempo. Não habilite a opção `Hidden` (oculto) pois vamos editar esse atributo.
4. Para visualizar o atributo criado, clique na guia `Attributes` posicionada à direita da guia `Pallete` , no canto inferior direito da tela.



| Name | Type | Hidden? |
|------------|--------|--------------------------|
| cronometro | Number | <input type="checkbox"/> |

Figura 5.1: Visualização de atributo criado

5. Vamos agora criar um evento para utilizar esse atributo. Clique em `+Add Event` ;
6. Vá até `Time` e selecione `Every N Seconds` . Nomeie-o para **Contagem**;
7. Vamos conectar ao bloco laranja o bloco para que possamos utilizar o atributo criado anteriormente. Na categoria `Numbers & Text` arraste o bloco `increment` da seção `Increment/Decrement` para dentro do bloco laranja;
8. Para que a contagem seja regressiva, substitua *increment* por *decrement* no bloco verde e insira **01** segundo no bloco laranja para iniciar a contagem;
9. No bloco verde, insira o valor **1** para que o cronômetro reduza a contagem nessa velocidade;



Figura 5.2: Instrução de contagem regressiva

10. Pronto! O bloco de contagem foi criado. Vamos anexar essa contagem à `Cena01`. Clique no botão verde `Attach to Scene` localizado à direita da tela, selecione `Cena01` e confirme;
11. É aqui que especificaremos o tempo de contagem para o atributo criado anteriormente, lembram? O tempo estará relacionado a várias questões como tamanho da cena, quantidade de desafios e obstáculos, por exemplo. Para o nosso jogo, vamos testar o tempo de **30** segundos.
12. Salve o jogo.

Pronto! Criamos um novo evento para que se inicie a contagem regressiva de 30 segundos no primeiro nível. Precisamos agora exibi-la na tela.

Exibindo a contagem regressiva na interface

Após criarmos a instrução para a contagem regressiva, precisamos exibi-la na interface do jogo para que o jogador tenha exata noção do tempo que tem para percorrer a cena, e para isso temos o evento `When Drawing`, utilizado para a exibição de informações na tela. Vamos aprender como utilizá-lo seguindo as etapas:

1. Vamos continuar no *scene behavior* `Contagem Regressiva`.
Selecione o evento `When Drawing`:

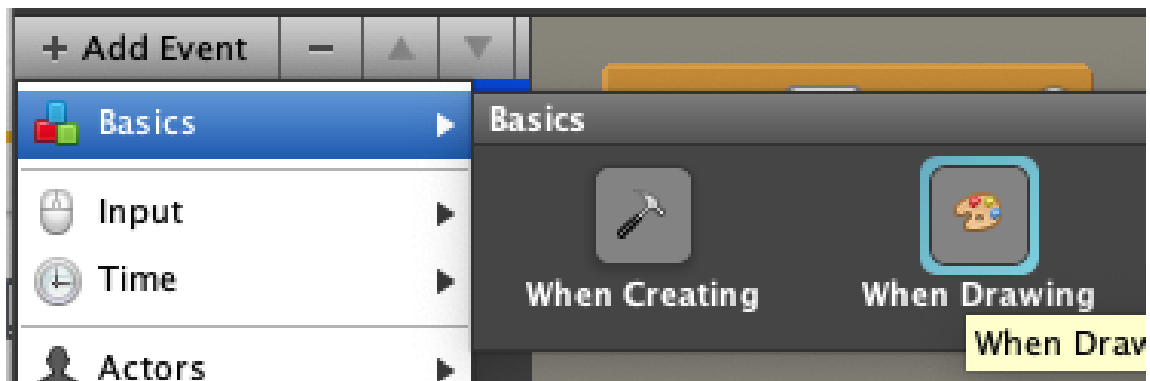


Figura 5.3: Seleção do evento When Drawing

2. Renomeie-o para **Exibir Contagem**;
3. Clique na categoria e na guia `Drawing` na `Palette`. Nesta categoria você encontrará opções para serem exibidas na interface do jogo;
4. Arraste o bloco `draw text` para dentro do bloco laranja. Ele será utilizado para a exibição de textos na interface;
5. Vamos exibir nossa contagem na tela posicionando-a a **10** pixels de distância do início da tela. Digite esse valor nos campos das coordenadas `x` e `y`;
6. Na categoria `Numbers & Text`, vamos clicar na guia `Text` e selecionar o bloco `__ & __` para relacionar duas instruções de textos;
7. No primeiro campo editável à esquerda do bloco verde digite a palavra **CONTAGEM**. Esse texto aparecerá antes da contagem regressiva na tela;
8. No outro campo editável do bloco verde, à direita, clique na seta de seleção e selecione `Basics`, `_ as text`, ou seja, algum atributo será convertido em texto para ser exibido na tela. Clicar na seta de seleção de blocos editáveis não difere de escolher as opções na `Palette`, a vantagem é que pelo campo editável só aparecerão opções que funcionem nessa instrução:

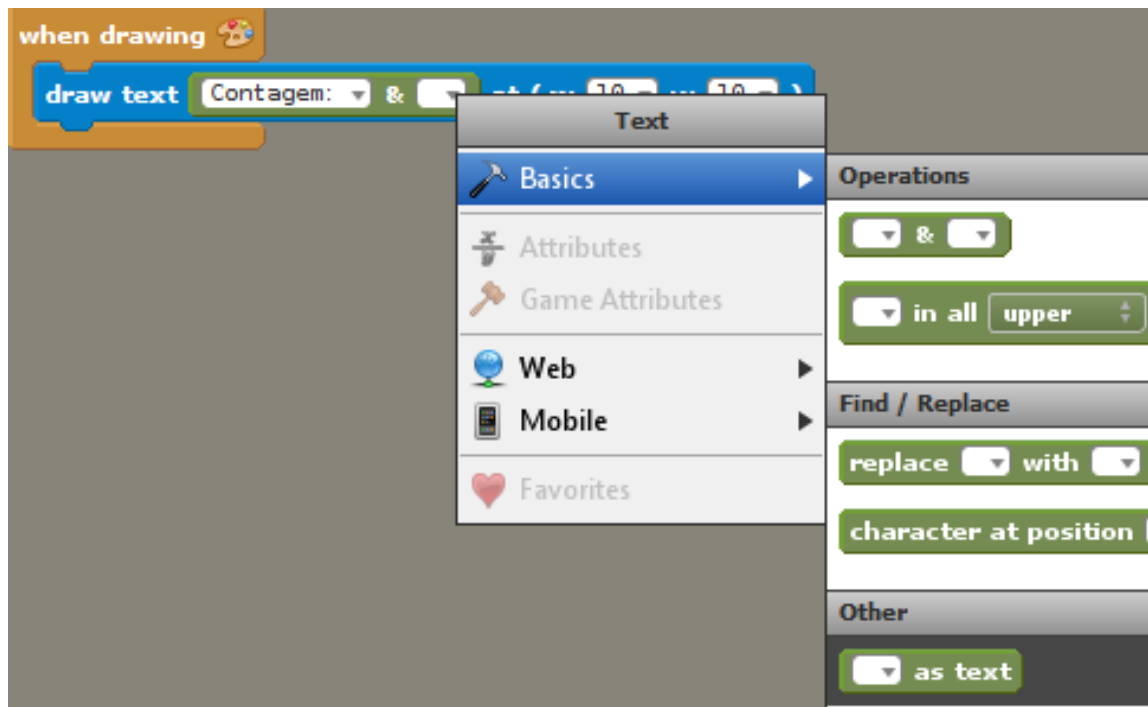


Figura 5.4: Seleção da instrução as text

9. No bloco `as text`, vamos inserir o atributo `cronometro`.
 Selecione-o na `Palette` em `Attributes` e arraste-o para dentro do bloco, conforme a seguir:



Figura 5.5: Instrução para exibição da contagem na interface

10. Salve e teste o jogo.

Repare que no canto superior à esquerda aparece a contagem visível na tela, com a fonte padrão do software. Sua formatação será realizada no próximo tópico.

DICA:

Caso queira reconfigurar o tempo da contagem regressiva após testar a cena completa, clique na guia `Behaviors` da cena correspondente.

Faça você mesmo!

Reposicione a contagem na tela alterando os valores das coordenadas *x* e *y*.

5.2 Formatando textos no Stencyl

Para melhorar o aspecto visual do texto na interface do jogo, é importante que formatemos a fonte de texto utilizada. Os textos incluídos na interface dependem dos arquivos de fonte instalados nos dispositivos digitais. Digitamos, por exemplo, na fonte *Arial* porque possuímos seu arquivo instalado no computador. Para que possamos formatar o texto inserido na interface do *Stencyl*, temos que inserir sua fonte no jogo. As fontes utilizadas no *Stencyl* devem ser do tipo *TrueType* que possuem a extensão `.ttf`, fontes mais adequadas para visualização em tela. Vamos baixar a fonte *Score Display* no `StencylForge` para seguir as etapas:

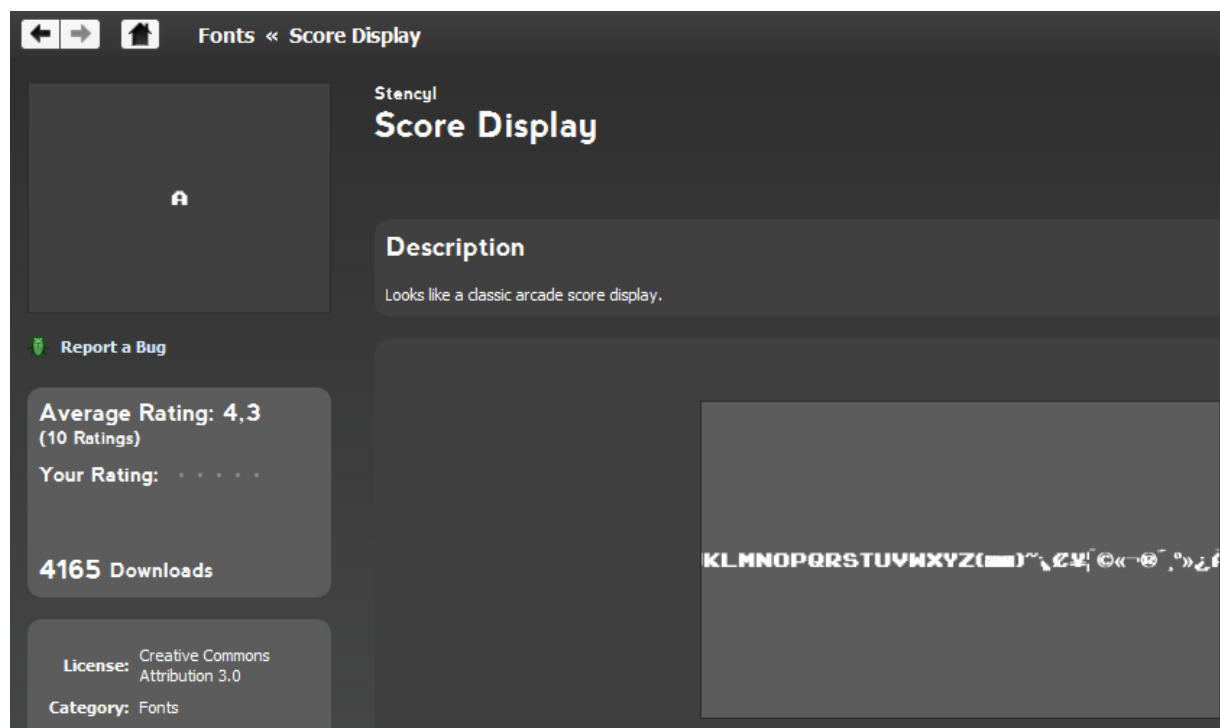


Figura 5.6: Tela de fonte no StencylForge

1. Clique em `Dashboard` e em `Fonts` para selecionar a fonte baixada;
2. Selecionaremos a fonte *Score Display*. Se você deseja instalar outras fontes, basta clicar em `Click here to create a new Font` , nomeá-la, clicar em `Create` e também seguir as próximas etapas;
3. No painel adiante, na guia `Font Style` , você encontrará várias opções para a escolha da fonte alterações no seu estilo:

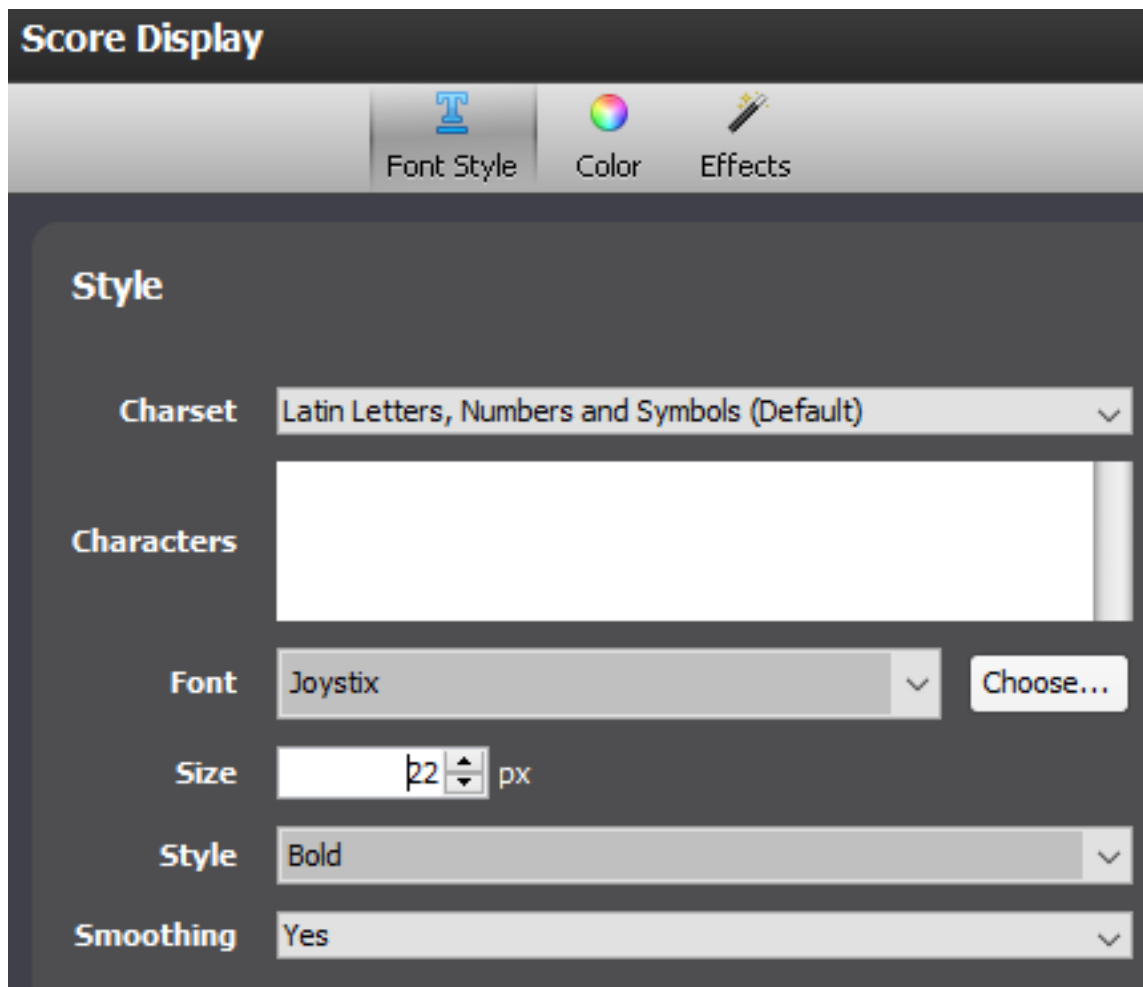


Figura 5.7: Opções para formatação da fonte

- **Charset**: codificação dos caracteres das fontes;
 - **Characters**: área customizável para exibição dos caracteres. Para acessá-la, selecione acima a opção `custom` ;
 - **Fonte**: localize à direita, no botão `Choose` , a fonte desejada para formatá-la;
 - **Size**: altere aqui o tamanho da fonte;
 - **Style**: altere aqui o estilo da fonte;
 - **Smoothing**: efeito de suavização da fonte na tela.
4. Na guia `Color` podemos escolher a cor ou as cores (gradiente) para nossa fonte:

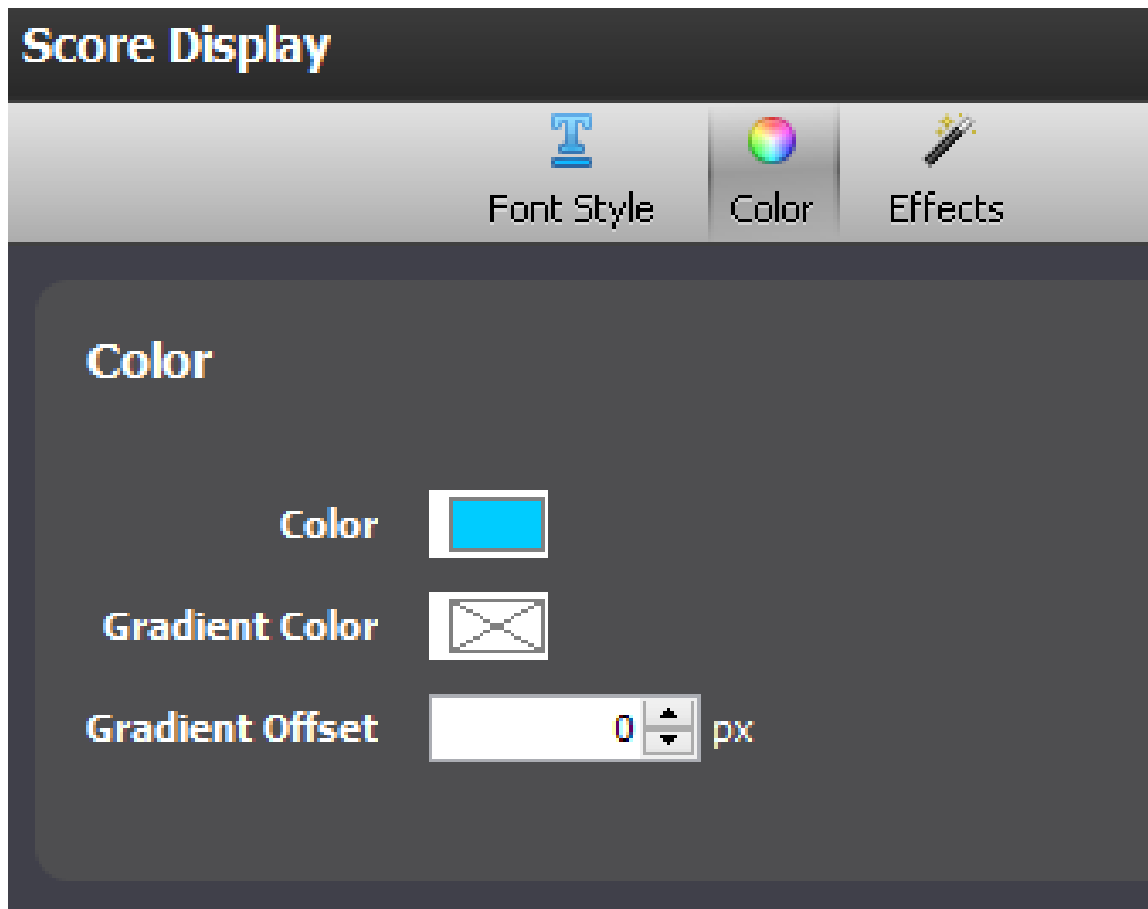


Figura 5.8: Opções para formatação da cor da fonte

5. Na guia *Effects* , podemos escolher a cor e a espessura do contorno da fonte (*Stroke*) e aplicar um efeito de sombreamento (*Shadow*):

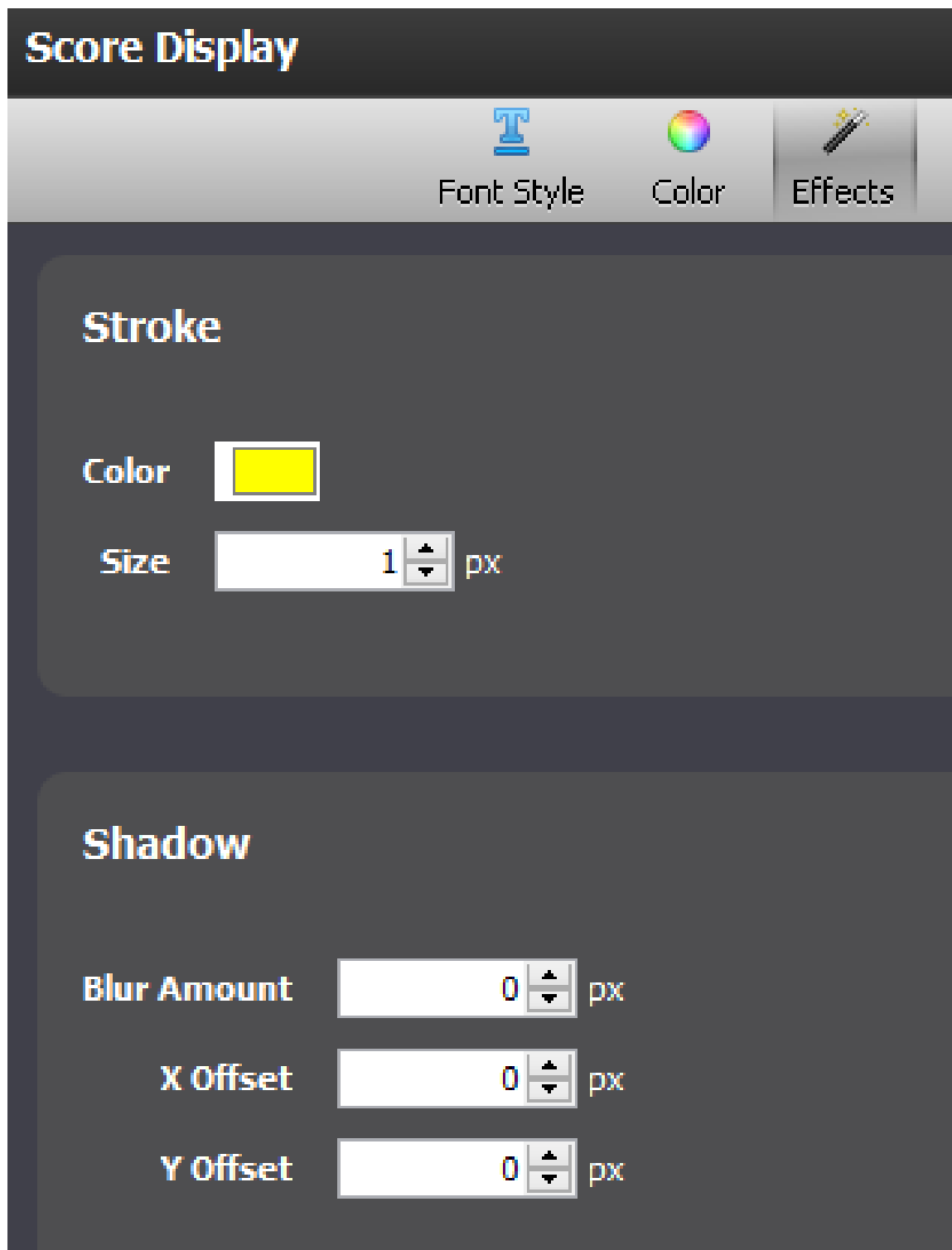


Figura 5.9: Opções para o contorno e sombreamento da fonte

6. Formate livremente sua fonte e salve o jogo.

Vinculando a fonte a uma instrução

Após ter incluído a fonte no *Stencyl* e tê-la formatado, vamos vinculá-la a uma instrução para que seja exibida na interface do jogo. Nós já temos uma instrução previamente criada para a contagem regressiva, lembram? Vamos acessá-la?

1. Volte ao **Dashboard** e selecione **Contagem Regressiva** ;
2. Clique no evento **Exibir Contagem** ;
3. Volte na opção **Drawing** em **Pallette** , mas desta vez clique em **Styles** . Nesta guia, você vai encontrar várias opções para formatar seu texto na interface do jogo;
4. Vamos escolher a fonte que será formatada na interface para o texto relativo à contagem regressiva. Arraste o bloco **set current font to** para dentro do bloco laranja e acima do bloco **draw text** . Veja:



Figura 5.10: Instrução para exibição de fonte na interface

5. Na caixa de seleção **Font** , escolha **Choose Font** para selecionar a fonte anteriormente formatada;
6. Salve e teste o jogo.

Veja que o texto da contagem regressiva está formatado conforme as alterações anteriores.

Faça você mesmo!

Agora é a sua vez! Formate livremente sua fonte conforme sua preferência. Não se esqueça de selecioná-la no painel **Dashboard** .

5.3 Inserindo uma barra para a contagem regressiva

Em vez de a contagem regressiva ser incluída na interface no formato texto, podemos inserir uma barra para a contagem do tempo. O procedimento é bastante simples, pois já criamos o atributo para a contagem e também já criamos o evento para que ela fosse exibida na tela. Então, vamos continuar utilizando o evento `Exibir Contagem` e trocar a exibição seguindo as etapas:

1. Clique com o botão direito no bloco azul `draw text` e selecione `Activate/Desactivate` para desativá-lo. Repare que sua opacidade diminui para mostrar que não está mais ativo na instrução.
2. A barra de contagem regressiva será visualizada por meio de um *shape* retangular. Vamos selecionar em `Drawing`, lá na `Pallette`, o bloco `draw rect at x:_ y:_) with (w:_ h:_)` para esse fim, e levá-lo para o bloco laranja abaixo do bloco `draw text`.
3. Configure-o conforme a seguir e teste o jogo para visualizá-la. Inserimos a opção `Fill` para que o *shape* retangular localizado nas coordenadas `x=10` e `y=10` com as dimensões `100` por `10` fosse preenchido.

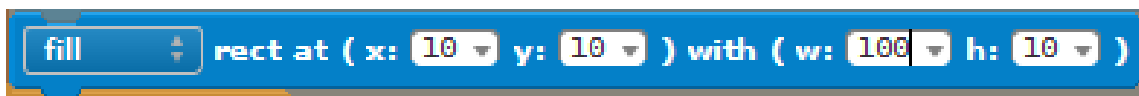


Figura 5.11: Bloco para exibição de shapes na tela

4. Você reparou que o preenchimento da barra está com a cor preta pois ainda não o formatamos e também ainda não inserimos o atributo de contagem regressiva. Vamos continuar?
5. Insira o atributo `cronometro` no campo `w` da barra `Fill` conforme adiante e teste o jogo.



Figura 5.12: Instrução para barra de contagem regressiva

6. Você deve ter reparado no teste do jogo que a barra está diminuindo, ou seja, ela já está funcionando, mas está pequena e sem formatação. Vamos deixá-la mais atraente. Primeiramente, vamos aumentá-la de tamanho. Tire o atributo `cronometro` do campo `w` e insira o bloco de instrução de multiplicação, encontrado em `Numbers & Text` e configure-o conforme a seguir, assim o tamanho padrão da barra será multiplicado por 5:



Figura 5.13: Instrução para redimensionamento da barra de contagem

7. Configure-o conforme desejar. Para concluir, vamos deixá-la mais atraente. Volte na opção `Drawing` na `Pallette` e clique em `Styles` ;
8. Nesta opção, encontramos várias configurações para a barra. Insira o bloco referente à cor da barra de preenchimento acima do bloco `fill rect` , escolha a cor, teste o jogo e veja como ficou.

Faça você mesmo!

Agora é sua a vez. Configure a barra de tempo livremente. Utilize a opção *Styles* em *Drawing* e também experimente outros valores para a largura e a altura. Veja o exemplo a seguir:

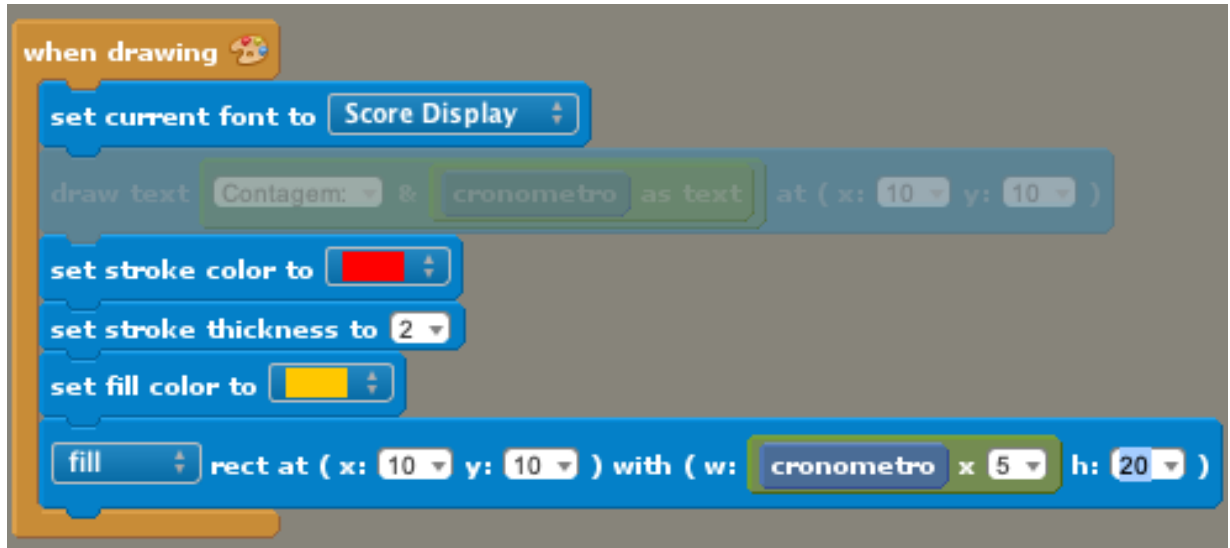


Figura 5.14: Instrução para a formatação da barra de contagem

5.4 Eliminando um actor quando a contagem terminar

Para encerrarmos este capítulo sobre contagem regressiva, vamos aprender a eliminar um *actor* do jogo assim que a contagem terminar. Para isso, conheceremos dois novos blocos de instrução bastante importantes no Stencyl: **if** e **for this scene**.

O bloco *if* refere-se à instrução condicional *se*, ou seja, alguma ação ocorrerá se algo acontecer. No nosso jogo, o *actor* Stencyl Book Monkey será eliminado e a contagem regressiva chegar a 0. O bloco *for this scene* é utilizado para especificar alguma instrução para determinada cena. Além disso, nesse último bloco, também aprenderemos a vincular a uma instrução um atributo criado para

um determinado *behavior*. Vamos entender melhor seguindo as etapas adiante:

1. Vamos criar um novo *actor behavior* chamado *Vida*;
2. Acrescente um novo evento `do after seconds` e mude seu nome para *Fim da contagem*;
3. Digite *01* em *seconds* no bloco laranja. Em 01 segundo o evento se iniciará;
4. Em `Flow`, na `Palette`, clique na guia `Conditions`. Vamos inserir o bloco de condição `if (se)`. Arraste-o para dentro do bloco `Every N Seconds`;
5. Vamos iniciar a condição *if* selecionando o bloco referente à igualdade/desigualdade na caixa de seleção, conforme a seguir:

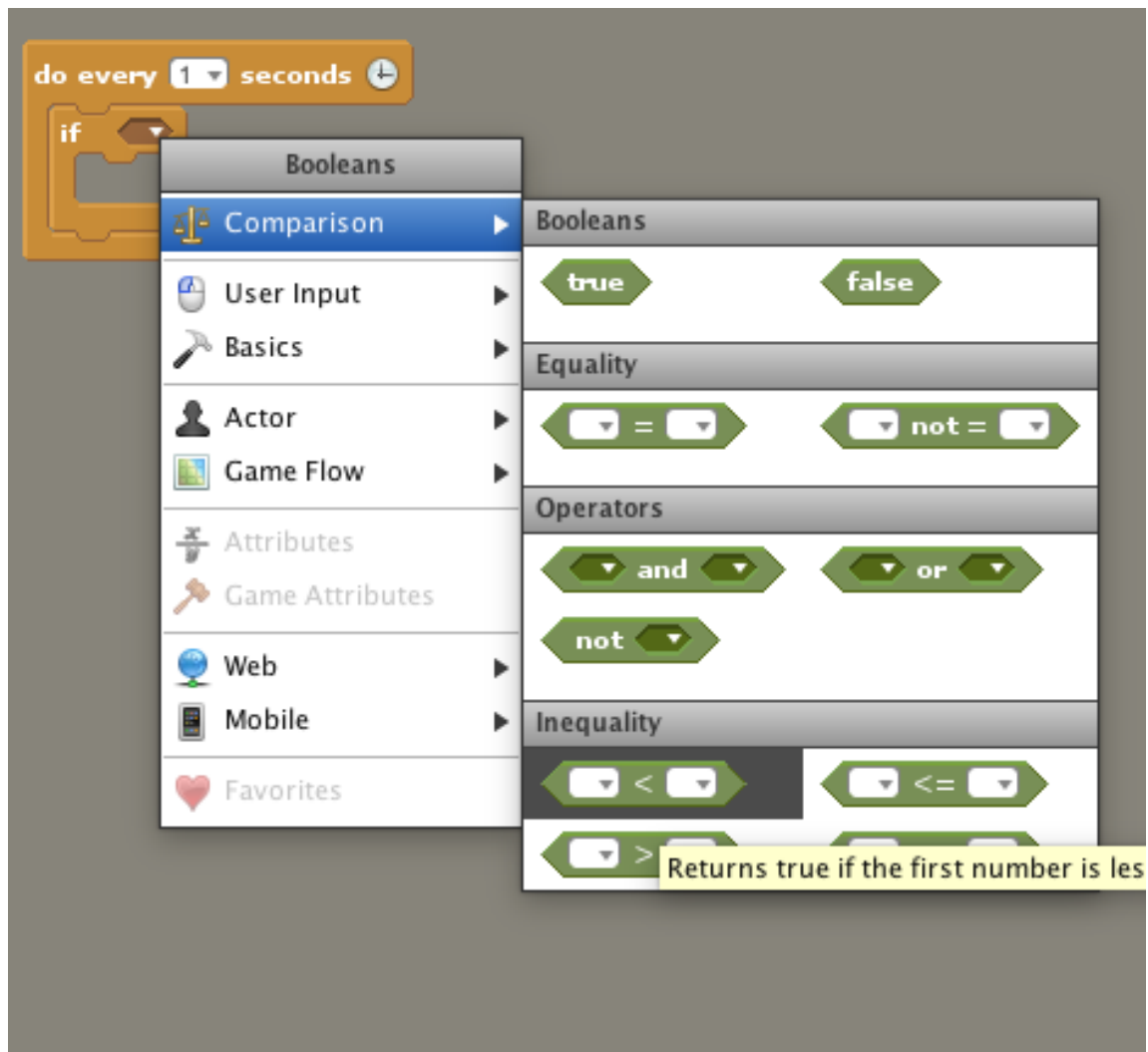


Figura 5.15: Seleção de bloco da instrução if

6. Vamos inserir no primeiro campo à esquerda do sinal < a instrução `for this scene` para podermos vincular a uma determinada cena um atributo de um outro *behavior*, no caso, o *behavior* contagem regressiva. Na guia *Palette*, clique no botão *Behaviors* e em *Attributes*;



Figura 5.16: Atributos da opção Behaviors

7. Arraste o bloco `for this scene, get _ from behavior` para dentro do bloco verde, para que possamos trazer o atributo `cronometro` para esta instrução:

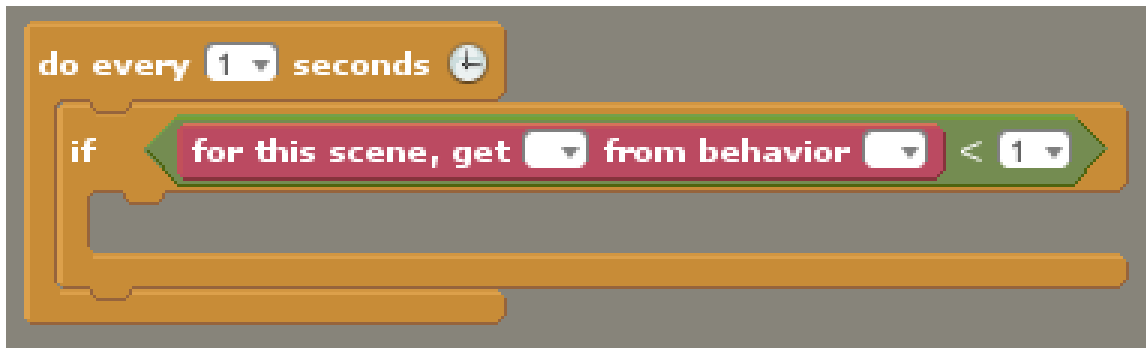


Figura 5.17: Configuração da instrução if

8. Para localizar o atributo `cronometro` do *behavior* de cena Contagem Regressiva , siga a orientação clicando na seta de seleção `get` :

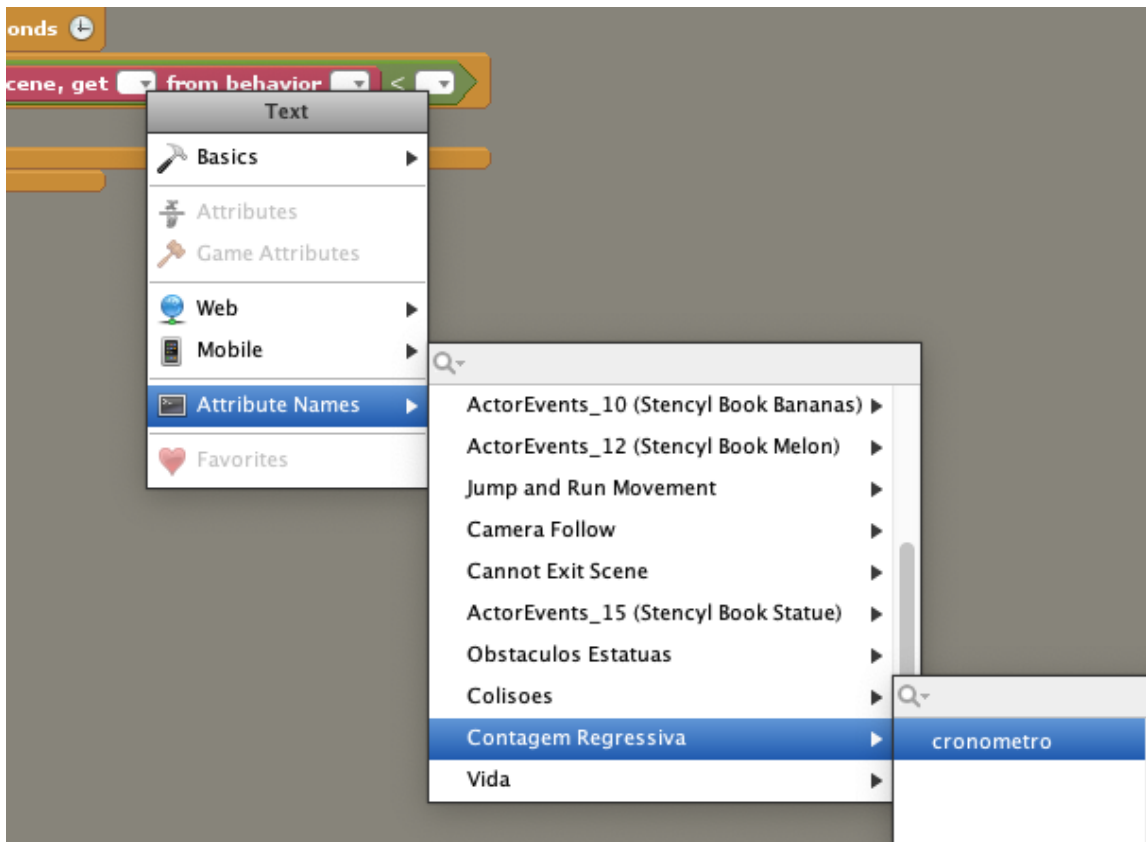


Figura 5.18: Seleção de atributo

9. Repare que automaticamente o campo `from behavior` foi preenchido com *Contagem Regressiva*, pois o atributo foi criado

para este *scene behavior*:

10. Para eliminar o *actor*, localize o bloco `kill Actor` e arraste-o para o bloco `if`. Veja abaixo:

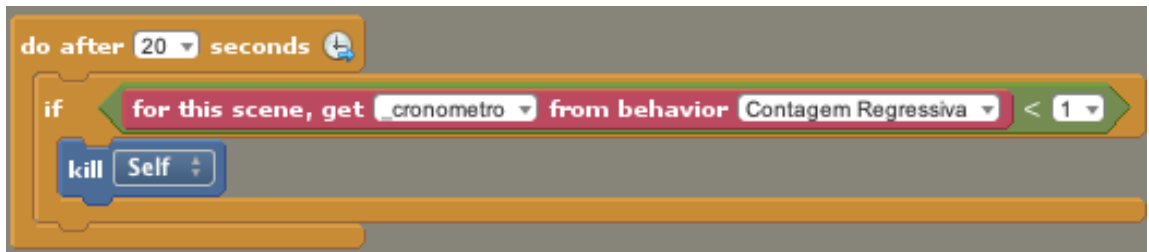


Figura 5.19: Instrução para eliminação de actor após contagem

11. Anexe este evento ao *actor* `Stency1 Book Monkey` ;
12. Teste o jogo.

Anexamos o *behavior* ao nosso herói e quando a contagem ficar menor que 01, ele sairá de cena.

Neste capítulo:

- aprendemos como cronometrar o tempo na cena;
- a exibir a contagem na interface;
- a exibir e configurar uma barra de tempo;
- a eliminar o *actor* do jogo após a contagem terminar.

No próximo capítulo, vamos nos aprofundar sobre como editar as animações do *actor*.

CAPÍTULO 6

Animações e efeitos visuais

Avançamos na mecânica do jogo inserindo mais desafios para o nosso personagem e agora vamos editar sua movimentação durante o jogo, sua animação. Uma animação é o estado do personagem de acordo com o seu momento no jogo, e dentro de uma animação podem existir um ou vários *frames*, dependendo da movimentação do personagem. Um *frame* é uma imagem referente a uma etapa da animação. Cada *frame* dessa animação 2D é conhecido como *Sprite*. Para compreendermos melhor, vamos dar uma olhada no exemplo a seguir referente à animação do nosso macaquinho:

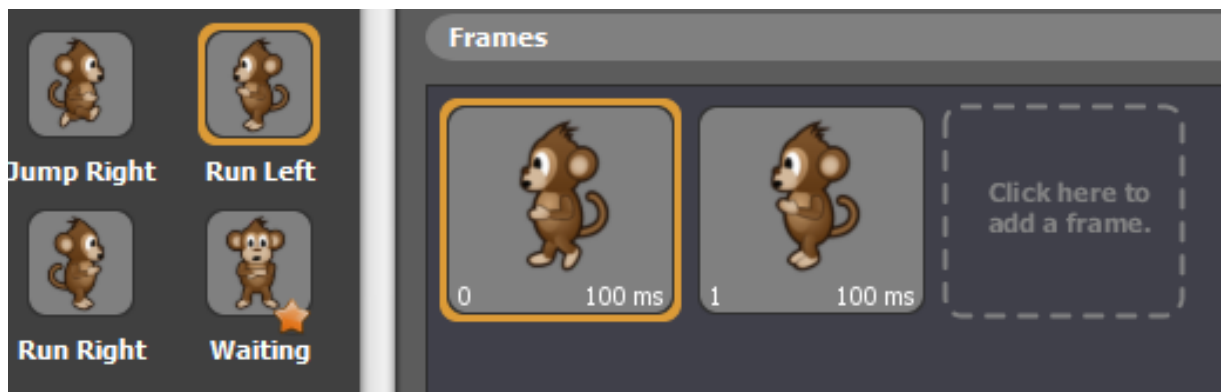


Figura 6.1: Frames de animação

À esquerda temos algumas das animações do nosso personagem. Ele pulando, deslocando-se para a direita ou para a esquerda. A animação *Run Left*, por exemplo, possui dois frames, demonstrados à direita. Isso quer dizer que, quando o macaquinho for controlado para se movimentar para a esquerda no jogo, dois *sprites* vão compor essa animação, dando-lhe movimento, e a velocidade dessa movimentação será medida em milissegundos (ms). Editaremos essa velocidade nos próximos tópicos.

6.1 Importando imagens para uma animação

Nosso jogo está sendo desenvolvido com as animações do próprio Stencyl sendo baixadas por meio do StencylForge, liberadas para uso por meio da licença *Creative Commons Attribution License* e criadas pela ilustradora Vicki Wenderlich. Para importar outros arquivos de imagem, softwares de tratamento de imagem como *Gimp* e de ilustração vetorial como *Inkscape* são ferramentas fáceis e eficientes para gerar esses arquivos que vão compor os *sprites* para as animações. Esses arquivos devem ser exportados para o *Stencyl* em `.png`, por ser um formato de imagem com fundo transparente.

Vamos importar uma nova imagem para o *Stencyl* que servirá posteriormente como mais uma animação para nosso personagem macaco. Antes, baixe a imagem `monkey_angel_actor` no link <https://github.com/scampelo/imgsBookStencyl>. Após baixar a imagem, siga as etapas adiante:

1. Vamos abrir a animação do nosso *actor* *Stencyl Book Monkey* em *Actor Types* ;
2. Vamos adicionar uma nova animação clicando no sinal de mais (+) no canto inferior esquerdo da tela;

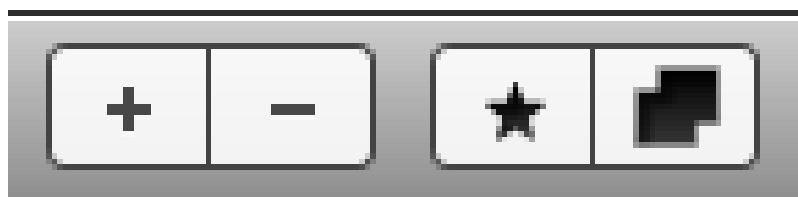


Figura 6.2: Opções do painel de animação

3. Repare que foi incluído um novo item entre as animações do nosso personagem, chamado *Animação 8*. Vamos incluir um *frame* para esta animação. Clique à direita em *Click here to add a frame* ;

4. Vamos conhecer as opções da janela a seguir antes de importarmos o arquivo:

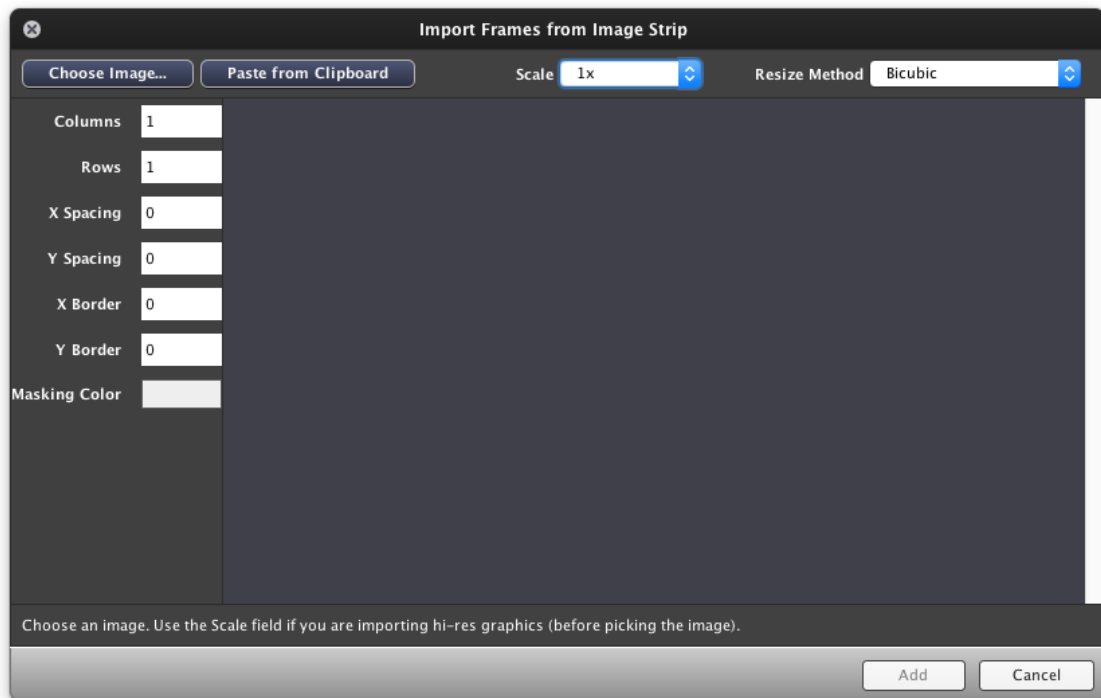


Figura 6.3: Janela de importação de imagem para frames

- *Choose image*: aqui localizaremos nosso arquivo;
- *Paste from Clipboard*: podemos colar aqui um arquivo que foi enviado para a área de transferência;
- *Scale*: escolha da resolução de imagem;
- *Resize Method*: opções de interpolação da imagem.

Coluna à esquerda:

- *Columns*: linhas divisórias verticais para arquivos com vários *frames*;
- *Rows*: linhas divisórias horizontais para arquivos com vários *frames*;
- *X Spacing*: espaçamento entre linhas verticais;
- *Y Spacing*: espaçamento entre linhas horizontais;
- *X Border*: bordas verticais;
- *Y Border*: bordas horizontais;

- *Masking Color*: cor de máscara para eliminação de fundo.
5. Vamos selecionar **1x** em *Scale* para a imagem vir com a resolução original (esta opção deve ser definida sempre antes da importação) e clicar em *Choose Image* para escolher o arquivo:
 6. Localize o arquivo `monkey_angel_actor` baixado e importe-o.

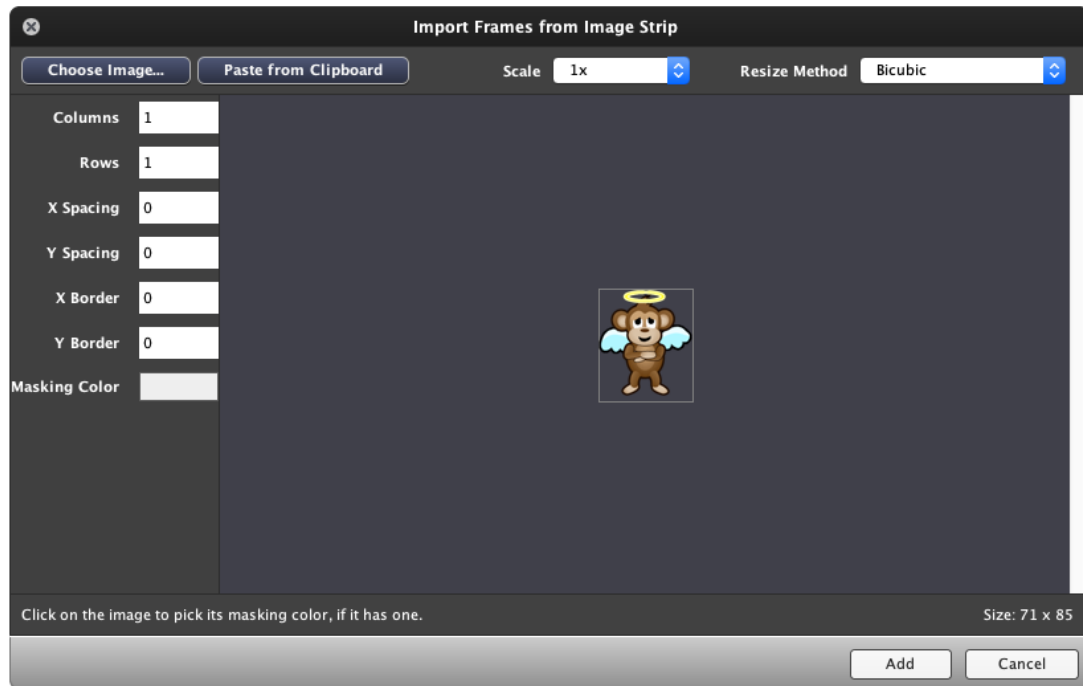


Figura 6.4: Visualização do frame importado

7. Vamos clicar em *Add* para adicionar essa imagem à *Animação 8* ;
8. No campo *Name* , altere o nome *Animation 8* para *Anjo*;
9. Essa animação substituirá a animação *Dead* . Remova-a selecionando-a e clicando no botão - (*Remove Animation*) no canto inferior esquerdo da tela.

Pronto! Inserimos uma nova animação para o `Stencyl Book Monkey` .

A animação que incluímos possui apenas um *frame*, mas algumas outras possuem 2 *frames*, como a *Run Left* e *Run Right*. São

apenas essas duas que aparecem animadas devido aos dois *frames* que possuem. Se clicarmos na animação *Run Left*, por exemplo, veremos os *frames* a seguir, um *frame* com os pés separados e outro com os pés juntos:



Figura 6.5: Frames de animação

Podemos editar a velocidade dos *frames* clicando duas vezes sobre ele. Após clicarmos duas vezes no *frame*, veremos que a duração em milissegundos padrão é 100. Quanto maior o valor, mais lento o *frame*.

Faça você mesmo!

Experimente valores maiores e menores para a duração do *frame* e veja o resultado.

6.2 Substituindo animações

Nas próximas etapas, aprenderemos a inserir uma animação diferente durante o jogo. No momento em que nosso macaquinho for sair de cena, vamos incluir uma instrução para que apareça a animação anteriormente incluída, chamada `Anjo`.

A animação *Anjo* deve aparecer toda vez que nosso macaquinho colidir com um inimigo ou quando a contagem regressiva terminar. Poderíamos inserir essa nova instrução nos *behaviors* *Colisões* e *Contagem regressiva*, mas o trabalho seria dobrado, não é mesmo? Para que não tenhamos que repetir a mesma instrução em vários eventos, vamos criar um evento personalizado para que ele possa ser "engatilhado" em outros. O bloco de instrução para personalizarmos eventos chama-se *custom event*:

1. Vamos criar este novo evento no *actor behavior* chamado *Vida*. Selecione-o no *Dashboard*;
2. Clique em *+Add Event*, desça até *Advanced* e selecione *Custom Event*;
3. Vamos mudar o nome deste evento para *Mostrar Anjo*;
4. No campo editável *when happens*, digite *MostrarAnjo*, sem espaço e com as iniciais maiúsculas. Vamos padronizar dessa forma para que, quando precisarmos selecionar essa instrução para que seja disparada em outro evento, não corramos o risco de digitá-la erroneamente. O nome do evento *Mostrar Anjo* com espaço não tem a mesma importância por se tratar apenas de um rótulo;
5. Na *Pallette*, no botão *Actors*, selecione a guia *Draw*;
6. Vamos arrastar o bloco *switch animation to* para dentro do bloco laranja. A função desse bloco é fazer a mudança da animação após uma determinada instrução;
7. A animação que queremos que apareça é a *Anjo*, do *Stencyl Book Monkey*. Digite corretamente o nome da animação, *Anjo*, no campo editável do bloco azul. Veja:

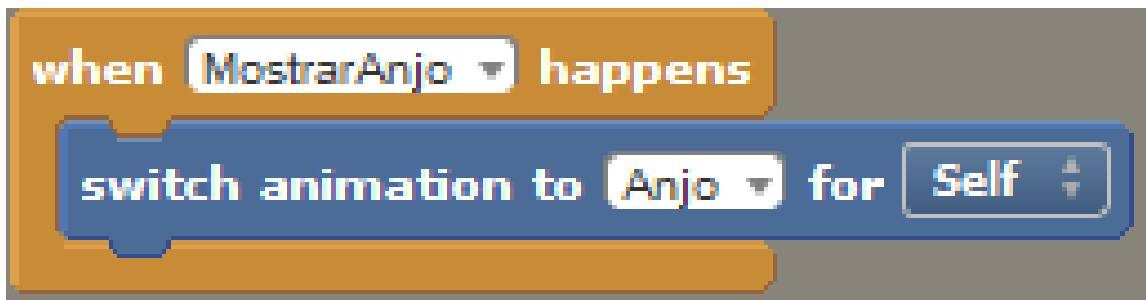


Figura 6.6: Bloco para substituição de animação

8. Em seguida, precisamos desabilitar os *behaviors* já incluídos anteriormente para esta animação. Na *Palette*, vamos clicar em *Behaviors* e em seguida na guia *Stage*;
9. Arraste o bloco *Enable Behavior Text for Actor* para dentro do bloco laranja e acima do bloco azul, conforme imagem a seguir, para desabilitar os *behaviors* inseridos anteriormente para a animação:



Figura 6.7: Bloco personalizado para substituição de animação

10. Vamos alterar *Enable* para *Disable* para desabilitar o *behavior* *Jump and Run Movement*:

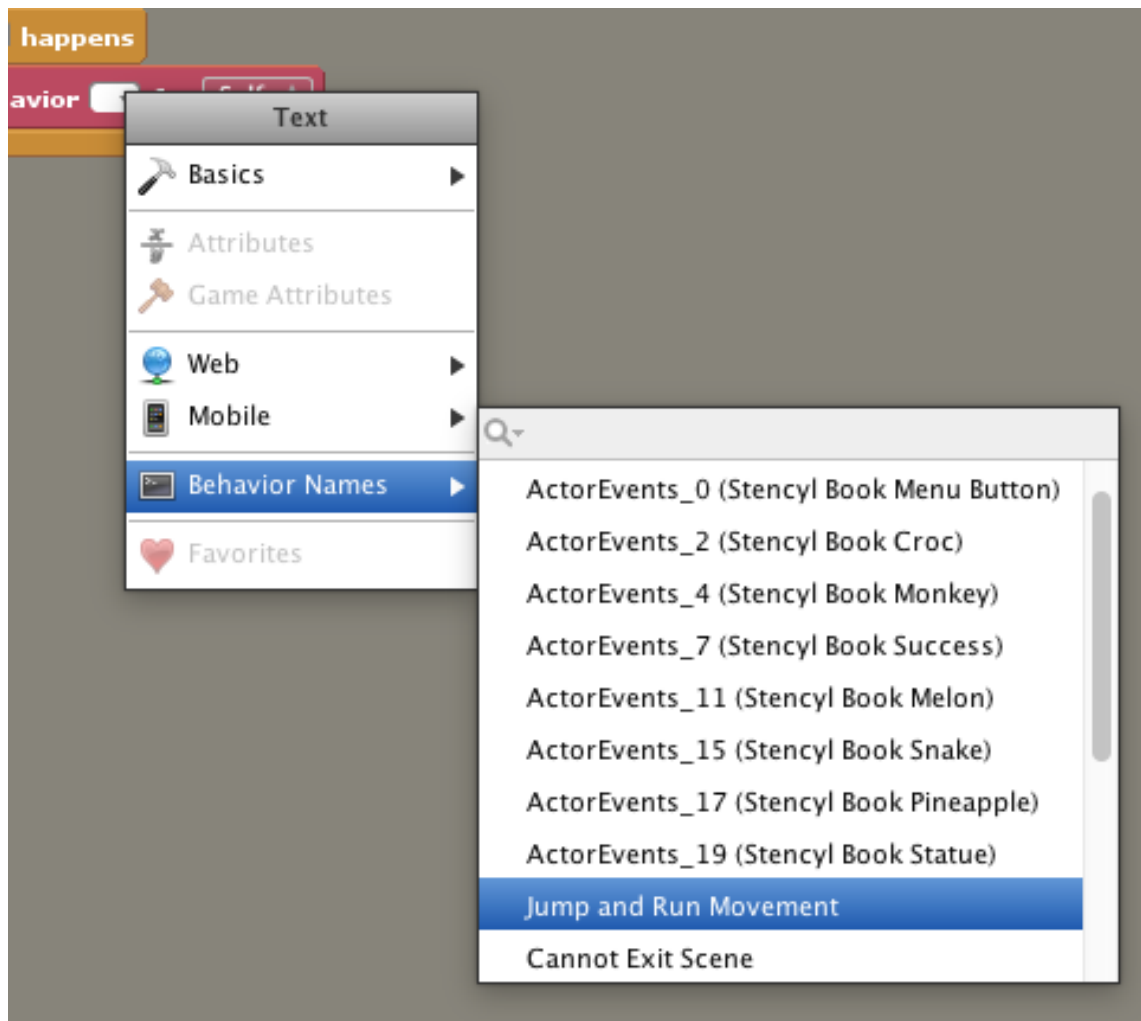


Figura 6.8: Procedimento para seleção de behaviors

11. Tire uma cópia do bloco vermelho arrastando-o para baixo mantendo pressionada a tecla `alt/option` e selecione `Colisoes` seguindo o procedimento anterior:



Figura 6.9: Configuração de bloco personalizado

12. Anexe este evento ao Stencyl Book Monkey clicando em Attach to Actor Type à direita da tela;
13. Teste o jogo.

Pronto! Concluímos o evento personalizado para que a animação Anjo apareça em outro evento. Agora vamos aprender a engatilhá-lo.

6.3 Engatilhando eventos personalizados

Para fazer com que um evento personalizado aconteça, utilizaremos o bloco `trigger event`. Vamos inserir primeiramente este bloco quando nosso macaquinho entrar em contato com alguns inimigos. Veja:

1. Vamos acessar o *actor behavior* Colisoos ;
2. Selecione o evento Colidir Inimigos ;
3. Vamos tirar o bloco `kill` do bloco laranja;
4. Encontraremos o bloco `trigger event in behavior for actor` fica na guia `Trigger`, dentro de `Behaviors` na `Pallette`. Arraste-o para dentro do bloco laranja conforme a seguir:

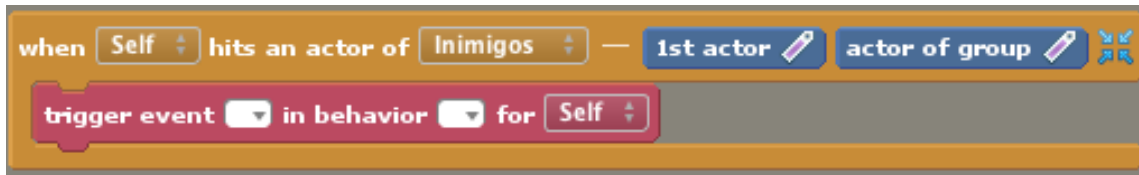


Figura 6.10: Bloco para engatilhar eventos

5. Digite na caixa de seleção à esquerda o nome do evento que queremos engatilhar, *MostrarAnjo* (sem espaços, da mesma forma que foi digitado anteriormente no evento personalizado);
6. Na caixa de seleção à direita, digite o nome *Vida, actor behavior* onde está localizado o evento *Mostrar Anjo*.
7. Teste o jogo.

Leve nosso macaquinho a colidir com um inimigo e veja que a animação *Anjo* aparecerá.

Substituindo a animação ao fim da contagem

Para inserir o macaco Anjo quando a contagem terminar, precisaremos criar um novo evento relacionado a este *actor*, pois é sobre ele que recairá também a mudança de animação, mesmo que a contagem seja feita para a cena. Vamos continuar com as etapas a seguir:

1. Vamos continuar no *actor behavior* *Vida*;
2. Vamos criar um novo evento de tempo *Every N Seconds*;
3. Renomeie-o para *Fim da Contagem*;
4. Digite **1** no campo editável *seconds* (segundos) para o evento acontecer neste prazo;
5. Vamos criar uma condição *if* para a contagem. Na *Pallete*, clique em *Flow* e arraste o bloco *if* para dentro do bloco laranja;
6. Ainda na guia *Flow*, vamos arrastar o bloco *0 0* para dentro do bloco *if*. A condição será para que se a contagem chegar a menor que 0, apareça o macaco anjo;

7. Vamos arrastar o bloco `for this scene get from behavior` para que possamos utilizar na cena em que o macaco estiver o atributo cronômetro do *scene behavior* Contagem Regressiva . Na Pallette , selecione Behaviors , clique na guia Attributes e arraste o bloco `for this scene get from behavior` para dentro da primeira caixa de seleção do bloco verde, conforme a imagem:



Figura 6.11: Bloco para utilização de atributos

8. No campo `get` , vamos selecionar o atributo `cronômetro` e digitar `1` para que a instrução aconteça abaixo dessa contagem;

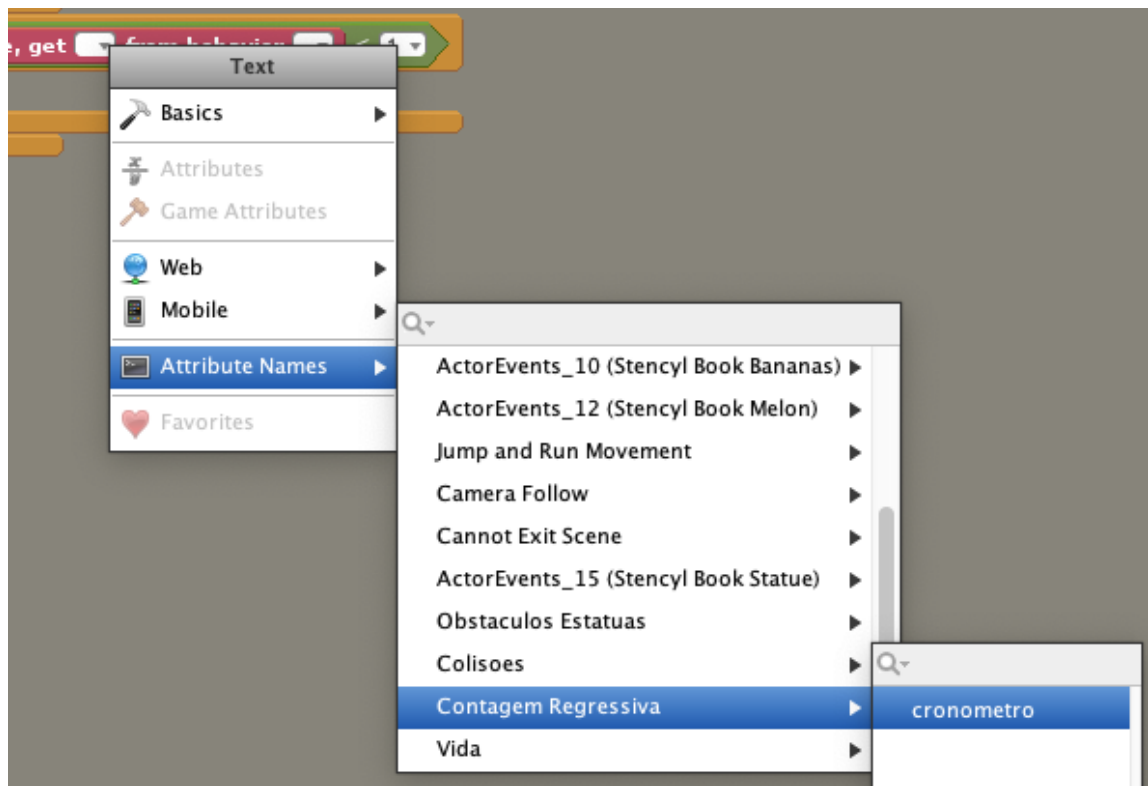


Figura 6.12: Etapas para seleção de atributos

9. Para concluirmos, vamos inserir o bloco `trigger event` para que o evento `Mostrar Anjo` seja engatilhado para o `Stencyl Book Monkey`. Digite o nome do evento, `MostrarAnjo`, e seu *actor behavior* correspondente, `Vida`. Veja como ficou:

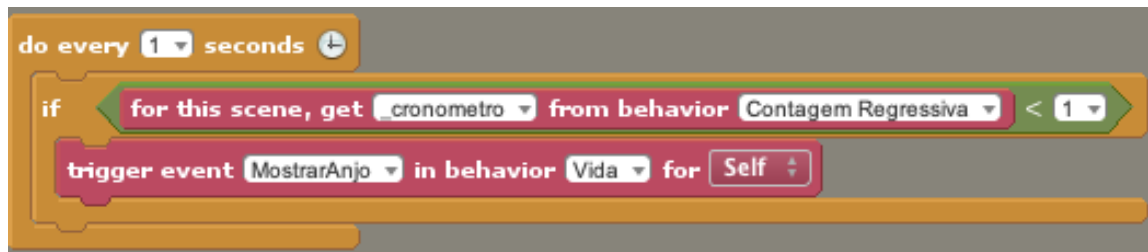


Figura 6.13: Bloco para o engatilhamento de eventos

10. Teste o jogo.

Agora o macaco anjo também aparecerá quando a contagem regressiva chegar a 0.

6.4 Efeitos visuais de animação

Para concluirmos este capítulo, vamos aprender como aplicar efeitos visuais na animação. São efeitos relacionados à alteração de cor e tamanho, por exemplo. Aplicaremos os efeitos na animação Anjo , portanto, após a mudança da animação acontecer. Vamos continuar, então, no evento Fim da Contagem . Vamos aos efeitos:

1. Na Palette , selecione a categoria Actors e clique em Tweening (interpolação) para escolhermos o efeito de animação;
2. Vamos aplicar o efeito grow (crescer) na animação Anjo . Arraste o bloco grow actor to do grupo Scale para baixo do bloco trigger event e configure-o conforme a seguir:

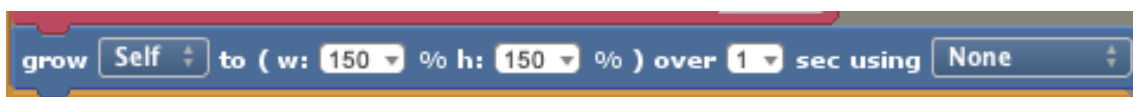


Figura 6.14: Bloco para efeito de redimensionamento do actor

3. O efeito grow será aplicado com o redimensionamento de 150% no prazo de 01 segundo;
4. Teste o jogo. O campo None possui efeitos para o redimensionamento da animação. Experimente para conhecê-los!

Na mesma categoria Actors , temos também efeitos relacionados à alteração de cor para nosso macaco anjo.

1. Vamos clicar na guia Effects ;
2. Para aplicarmos o efeito de cor, precisamos primeiramente arrastar para a instrução o bloco apply effect to . Solte-o abaixo do bloco grow to dentro do bloco laranja;
3. Um pouco mais abaixo do bloco apply effect to , temos alguns efeitos para alteração de cor. Como exemplo, vamos deixar nosso anjinho amarelado. Arraste o bloco make sepia para dentro do bloco apply effect .

4. Teste o jogo.

DICA:

Para remover o efeito, você pode inserir um bloco `Time` abaixo do efeito e incluir o bloco `remove all effects`.

Faça você mesmo!

Experimente outros efeitos. Veja outras opções:

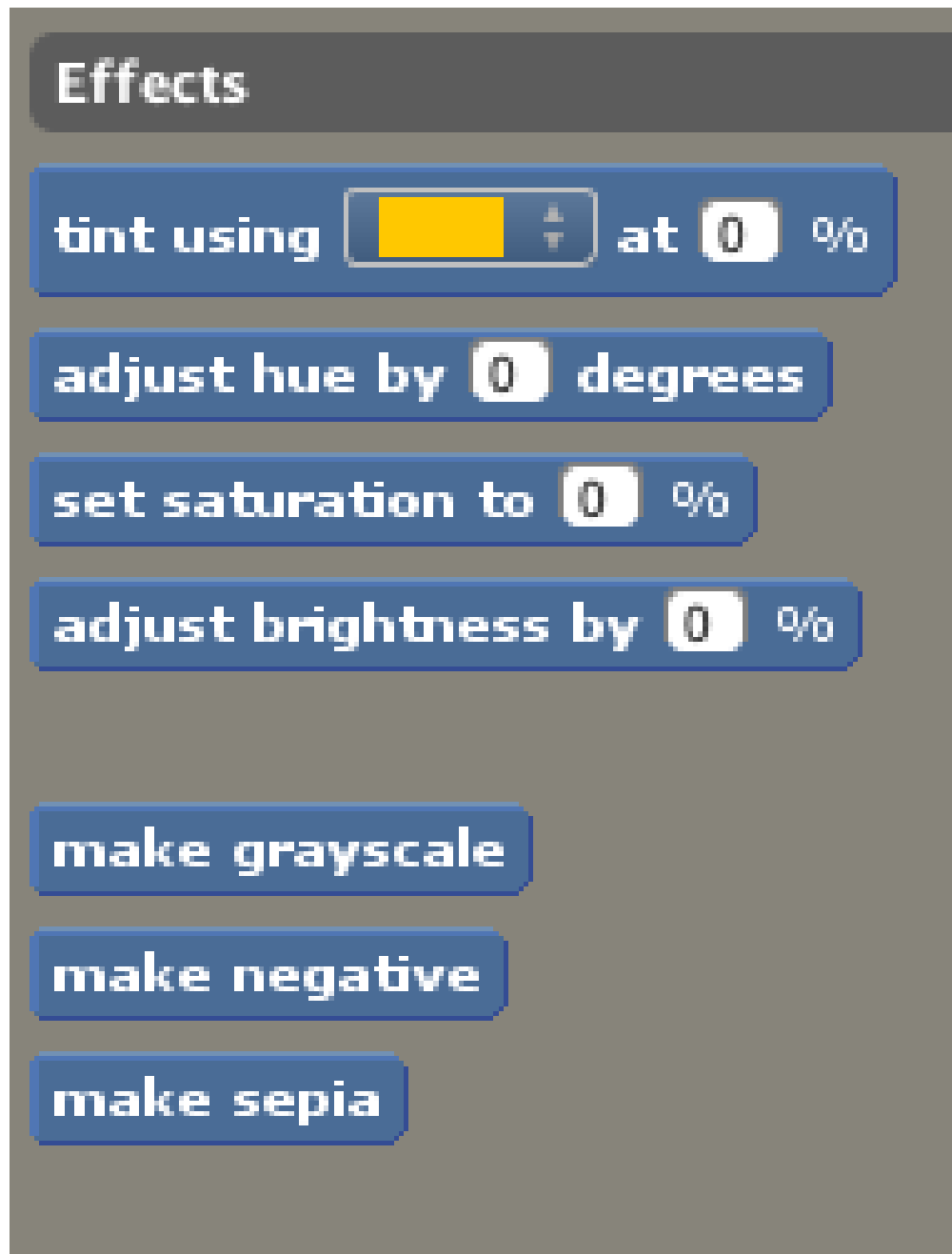


Figura 6.15: Efeitos para alteração da cor do actor

Neste capítulo:

- aprendemos a importar animações;
- a substituí-las no jogo;
- vimos como aplicar efeitos visuais.

No próximo capítulo, vamos finalizar os aspectos de jogabilidade, criando mais vidas e novos níveis.

CAPÍTULO 7

Vidas e Game Over

Nosso jogo está cada vez melhor, não é verdade? Neste capítulo, vamos trabalhar com as vidas do nosso *actor*, dando-lhe mais chances para enfrentar seus desafios. Além disso, após concluídas as chances, vamos desenvolver instruções para um *game over*, o fim de jogo. Mas, para isso, vamos precisar aprender a reiniciar a cena após o fim da contagem ou quando nosso *actor* colidir com um dos seus inimigos e, depois, dar-lhe mais vidas para continuar.

7.1 Reiniciando a cena

Reiniciando a cena, nosso *actor* poderá tentar completar o nível mais uma vez, ou quantas vezes desejarmos. Os obstáculos que nosso macaquinho tem no seu caminho são de inimigos e de tempo, e para não modificarmos duas instruções, a de contagem regressiva e a de colisão com os inimigos, vamos utilizar o evento `Mostrar Anjo` que criamos anteriormente e que já foi engatilhado nos dois outros eventos. Modificando-o, alteraremos também os outros dois.

1. Vamos seleccionar o evento `Mostrar Anjo` criado no *actor behavior* `Vida`.
2. Na `Palette`, vamos clicar em `Scene` e, em seguida, na categoria `Game Flow`. Nesta categoria encontraremos blocos para o bom fluxo do jogo como transições e recarregamento de cenas.
3. Para recarregar a cena, vamos arrastar o bloco `reload and fade out for secs...for secs using fade in` para dentro do bloco laranja, conforme a seguir:

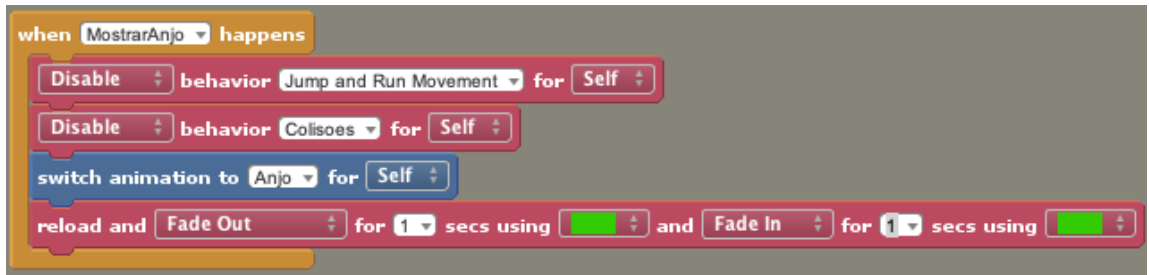


Figura 7.1: Instrução para o recarregamento da cena

4. Veja a configuração: a cena vai desaparecendo por **01** segundo na cor escolhida na caixa `secs using` e voltará também por **01** segundo na cor escolhida na última caixa. Escolha as cores clicando nas caixas e em seguida em `Choose Color`.

5. Teste o jogo.

Colida com um dos inimigos e também espere a contagem chegar a **0** para ver se a cena será recarregada. Depois, experimente outros valores e cores.

7.2 Criando mais chances ao actor

Após criarmos a instrução para o recarregamento da cena, vamos criar o evento para dar mais vidas para nosso macaco, para depois desenvolver a instrução para o *game over*. Por enquanto, após a instrução que acabamos de fazer, a cena está se recarregando infinitamente. Precisamos definir uma quantidade de vezes para limitar as chances do nosso *actor* para finalizar o nível e para que, caso ele não consiga atingir seu objetivo de coletar todas as frutas para avançar de nível, o jogo termine.

Primeiramente, vamos criar um novo atributo para o jogo (*game attribute*). Diferente do atributo para *behaviors* criado anteriormente, os *game attributes* são carregados no jogo e podem ser utilizados em todo o jogo, incluindo os seus vários níveis. A configuração de

novas vidas para nosso macaquinho servirá também nos próximos níveis do jogo, portanto, em todo o jogo. Vamos criá-lo seguindo as etapas adiante:

1. Vamos clicar na opção `Settings` (configurações) na parte superior da tela;
2. No painel à esquerda, clique em `Attributes` ;
3. Como ainda não temos nenhum atributo de jogo criado, vamos clicar em `Click here to create a Game Attribute` ;
4. Digite o nome *Vidas* para o atributo e mantenha a categoria padrão do Stencyl (Default);
5. O tipo de atributo que utilizaremos é o `Number` , pois trata-se da contagem das vidas. Clique nesta guia e digite **3** no campo abaixo. Essa será a quantidade de chances que nosso macaquinho terá até o *game over*;
6. Vamos confirmar em `ok` ;
7. Vamos selecionar o evento `Mostrar Anjo` no *actor behavior* *Vida* ;
8. Agora vamos iniciar a criação de uma condição: se o atributo de jogo *Vidas* for maior do que **0**, continue recarregando a cena. Encaixe o bloco `if` abaixo do bloco `switch animation` e inclua o bloco `reload` dentro do bloco `if` . Veja:

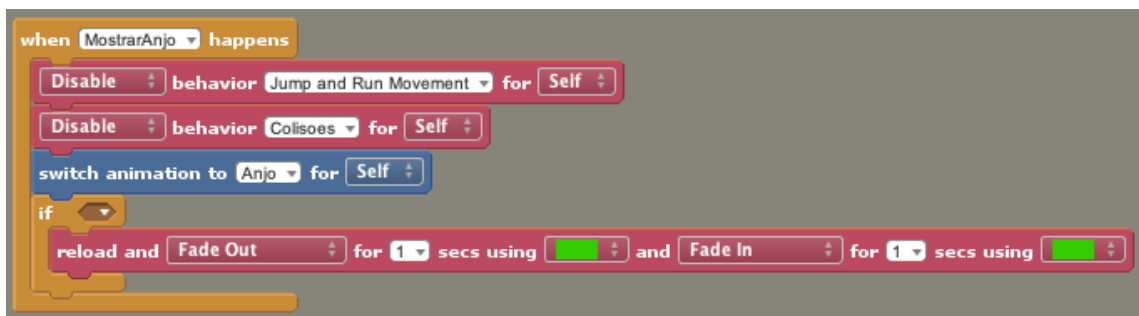


Figura 7.2: Condição para o recarregamento da cena

9. Vamos inserir o bloco de comparação maior que () dentro do bloco `if` para que, se as vidas estiverem acima de **0**, a cena continue sendo recarregada. Clique na categoria `Flow` e arraste o bloco maior que () para dentro do bloco `if` ;

10. Vamos inserir o atributo de jogo `Vidas` na caixa de seleção à esquerda. Clique na categoria `Game Attributes` e, na guia `Getters`, onde encontramos os atributos criados, e arraste-o para dentro da caixa de seleção à esquerda. Digite `0` na caixa à direita. Veja a seguir:

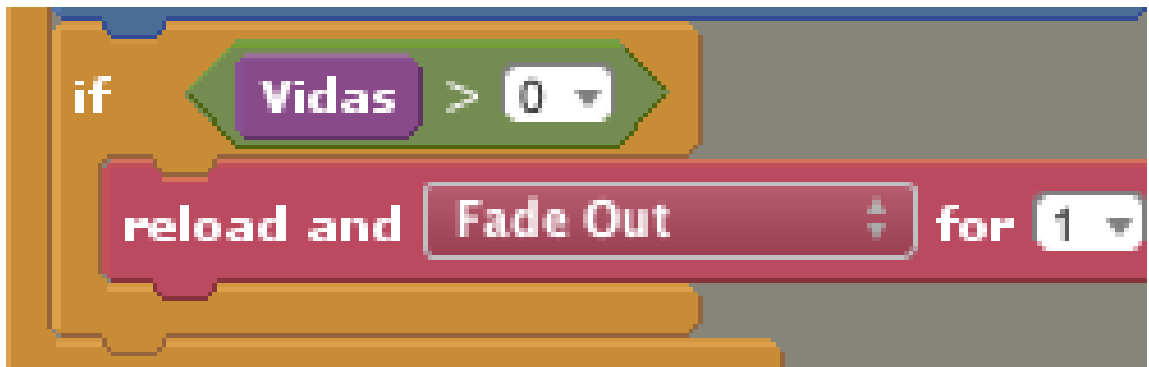


Figura 7.3: Atributo e bloco de comparação

11. Vamos inserir o bloco para configurar a contagem das vidas. Selecione a categoria `Game Attributes`, clique na guia `Setters` e arraste o bloco `set Vidas to` para dentro do bloco `if`. O bloco `Setters` é utilizado para definir valores para um atributo. Veja:



Figura 7.4: Bloco para configuração de valores para atributos

12. Arraste o bloco de subtração localizado em `Numbers & Text` para dentro do bloco `set Vidas to`;



Figura 7.5: Bloco para subtração

13. Vamos completar o bloco `set Vidas to` inserindo o atributo de jogo `Vidas` na caixa de seleção à esquerda. Clique na categoria `Game Attributes` e, na guia `Getters`, arraste-o para dentro do bloco de subtração no campo à esquerda;
14. Digite **1** na caixa à direita. Dessa forma, a variável será reduzida sempre em 1 vida:



Figura 7.6: Instrução para redução da contagem de variável

15. Teste o jogo.

Colida com inimigos e espere a contagem terminar. Verifique se a cena recarrega por duas vezes. Na próxima seção, vamos continuar a condição `if` para que, se as vidas chegarem a **0**, o jogo termine.

Faça você mesmo!

Insira a contagem de vidas na tela do jogo. Tente realizar seguindo o exemplo:

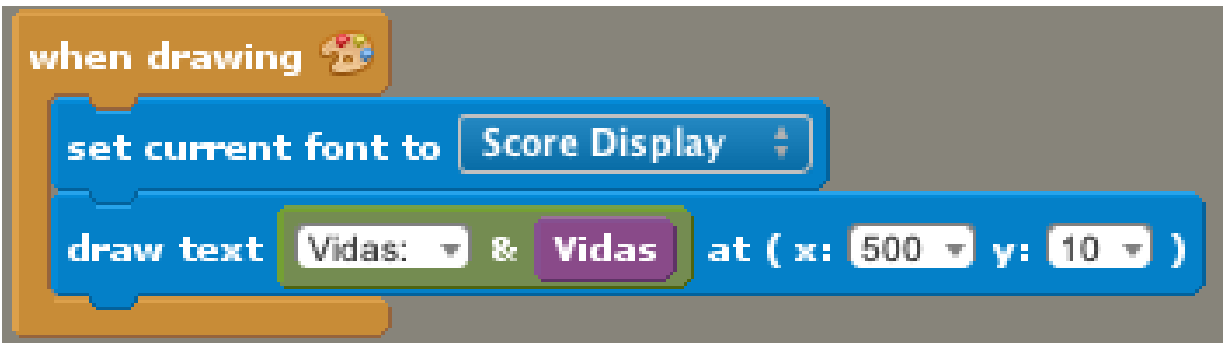


Figura 7.7: Instrução para exibição de variável na tela

7.3 Game Over

Após aumentar as vidas do nosso *actor*, vamos terminar o jogo após suas chances acabarem. Para isso, seguiremos no evento *if* para, partindo da primeira condição, criar uma nova instrução caso a primeira não aconteça. No nosso exemplo, se as vidas não forem maiores do que **0**, queremos que o jogo seja interrompido.

Para orientar nosso jogador de que o jogo acabou, vamos inserir uma mensagem na tela do jogo: *Game Over*. No Stencyl, essas mensagens devem ser incluídas como *actors* para que possamos criar as instruções posteriormente. São arquivos de imagens que serão incluídos nos eventos e instruções, assemelhando-se aos outros *actors* utilizados, como nosso macaquinho e as frutas, por exemplo. Então, vamos baixá-lo no StencylForge?

1. Localize no StencylForge o arquivo Stencyl Book Game Over Banner e baixe-o;
2. Abra-o em Actor Types , no Dashboard , e, na guia Physics , desative sua movimentação em Cannot Move ;
3. Abra o *actor behavior* Vida no Dashboard ;
4. No evento Mostrar Anjo , vamos completá-lo com a condição "caso contrário" (*otherwise if*), para que, quando as chances do nosso herói terminarem, apareça a mensagem *Game Over*. Na

categoria `Flow` , arraste o bloco `otherwise if` e inclua-o abaixo do bloco `if` e dentro do bloco `when` . Veja:



Figura 7.8: Instrução de bloco da condição otherwise if

5. Na caixa de seleção `if` , insira o bloco de comparação menor ou igual `=` ;
6. Mantendo pressionada a tecla `alt/option` , mova o atributo `Vidas` para dentro da caixa de seleção à esquerda e digite, à direita, o número **1**, ou seja, quando as vidas forem menores do que 1, a mensagem aparecerá:

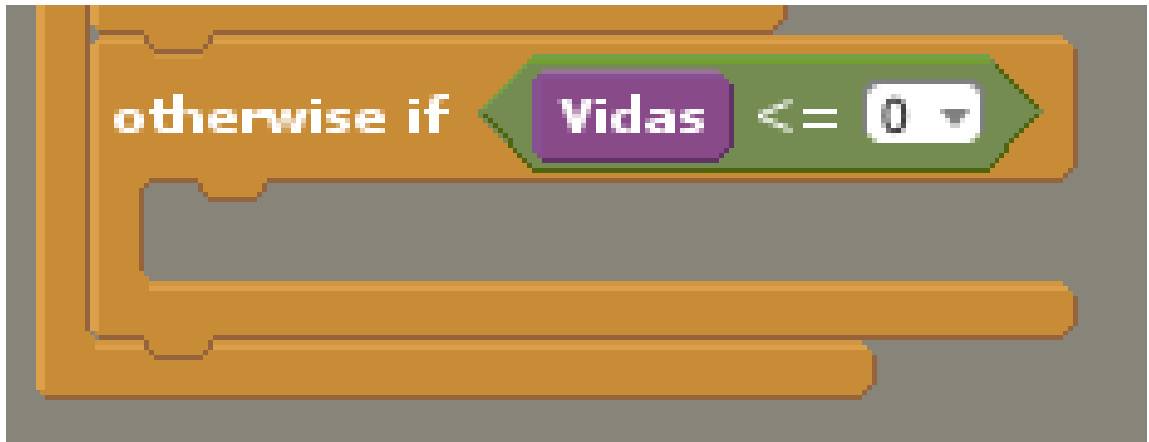


Figura 7.9: Bloco de condição otherwise if

7. Após inserir o bloco de condição, vamos inserir o *actor* `Stencyl Book Game Over Banner` na interface. Selecione o bloco `create actor type at x y` na categoria `Scenes` e arraste-o para dentro do bloco `otherwise if` :



Figura 7.10: Bloco para criação de actors na tela

8. Selecione o *actor type* `Stencyl Book Game Over Banner` na caixa de seleção `create` ;

9. Nossa tela de dispositivo para o jogo foi configurada com as dimensões 640 por 480 pixels, lembra? Vamos configurar o restante do bloco para que a mensagem fique centralizada na tela e acompanhe a câmera do dispositivo. Vamos inserir o bloco de soma, e somar 160 pixels na horizontal a partir do x (0) da câmera. Veja como ficou:

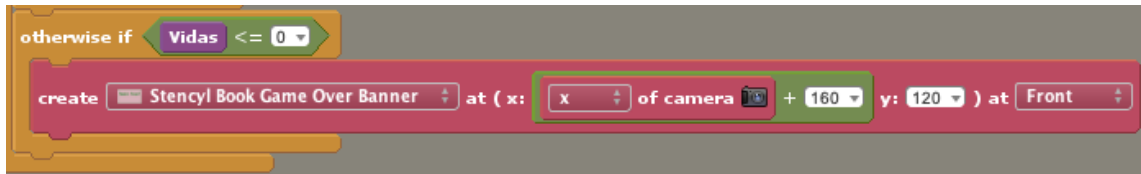


Figura 7.11: Instrução de posicionamento do actor na tela

10. Para concluirmos, vamos pausar o jogo para o aparecimento da mensagem. Na guia *Game Flow* da categoria *Scenes*, arraste o bloco *pause game* e insira-o acima do bloco *create*;

11. Teste o jogo.

Após o término das três chances, a mensagem *Game Over* aparecerá na tela do jogo.

7.4 Calculando os itens colecionáveis

Após trabalharmos com as vidas do nosso herói e com o *game over*, vamos desenvolver instruções para que ele colete as frutas da tela para que, atingindo esse objetivo, consiga avançar de nível. A instrução segue a estrutura do evento para a contagem das vidas. Criaremos primeiramente o atributo para a contagem das frutas para, posteriormente, vinculá-lo ao evento. Para criarmos o atributo, vamos relembrar seguindo as etapas:

1. Vamos criar um novo *scene behavior* chamado *Frutas Coletadas*;

2. Na *Palette*, vamos criar um novo atributo do tipo *Number* para a contagem com o nome *Frutas*, clicando em *Attribute* e em seguida em *Create an Attribute*;
3. Vamos criar um novo evento, *when creating*, encontrado em *Basics*. Esse evento será carregado uma vez sempre que o *behavior* *Frutas* for iniciado;
4. Renomeie-o para *Coletar Frutas*;
5. Na categoria *Scenes*, na guia *Actors*, arraste o bloco *for each actor group* para dentro do bloco *when created*. Especificaremos neste bloco o que acontecerá com cada *actor* do grupo *Frutas*;
6. Vamos selecionar o grupo *Frutas* em *Actor Group*;
7. Vamos clicar na guia *Numbers & Text* e arrastar o bloco *increment* para o aumento da contagem das frutas para dentro do bloco *for each*:



Figura 7.12: Bloco increment incluído na instrução

8. Vamos incluir nesta instrução o bloco *set actor _ for_ to*. Utilizaremos esse bloco para configurar um determinado valor para um *actor* ou um grupo de *actors*. Neste caso, vincularemos ao grupo *Frutas* valores como falso ou verdadeiro. O bloco *set actor _ for_ to* está localizado na categoria *Actors* e na guia *Properties*. Arraste e insira-o diretamente abaixo do bloco *increment*;

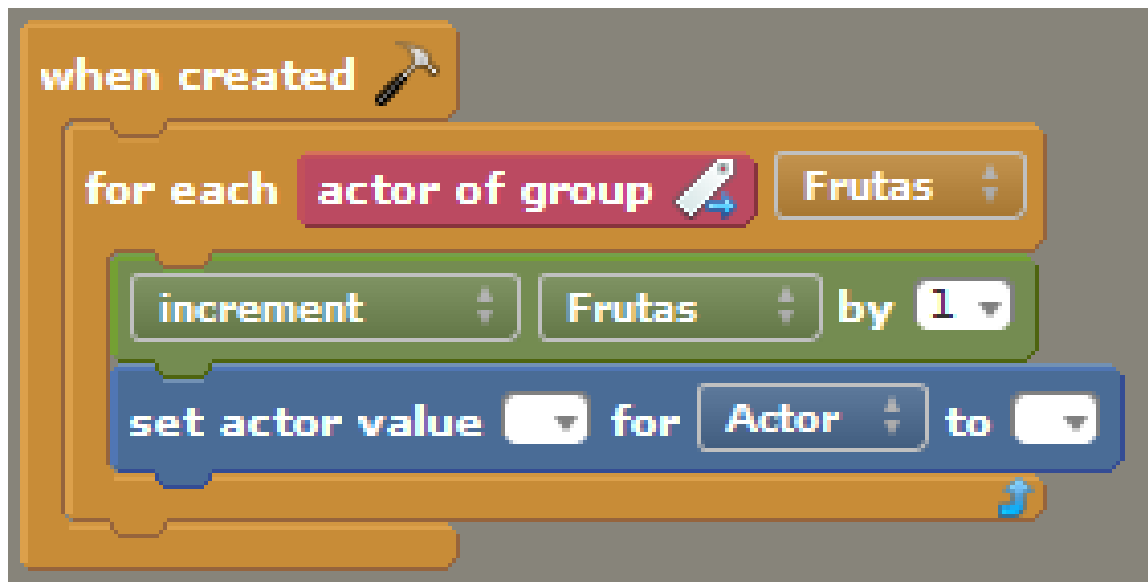


Figura 7.13: Inclusão do bloco set actor to na instrução

9. Vamos nomear esse valor como *Coletadas*. Copie o bloco vermelho para o bloco *actor* e, na caixa de seleção *to*, insira o valor **false** encontrado na categoria *Flow*. Dessa forma, atribuímos um valor para os *actors* do grupo *Frutas*:



Figura 7.14: Configuração do bloco set actor value

Até aqui, configuramos o valor *coletadas* como **false** para a coleta de cada fruta de 1 em 1. Vamos agora criar um evento para quando todas as frutas forem coletadas, e quando isso acontecer, vamos exibir uma mensagem de êxito para nosso jogador. Antes de continuarmos, vá lá ao *StencylForge* e

baixe o *actor* Stencyl Book Success . Não esqueça de confirmar se a opção *Cannot Move* está ativada na guia *Physics* do *actor*. Depois, vamos continuar com as etapas a seguir:

10. Vamos personalizar um evento para que, quando todas as frutas forem coletadas, apareça a mensagem de êxito. Em *+ Add Event* , vamos descer até *Advanced* e clicar em *Custom Event* . Insira também esse nome, sem espaço, no bloco amarelo:



Figura 7.15: Bloco de personalização de eventos

11. Arraste o bloco *increment* em *Numbers & Text* e configure-o conforme a seguir para que a contagem das frutas seja feita de forma decrescente de 01 em 01:



Figura 7.16: Bloco para o aumento da contagem

12. Vamos inserir o bloco *if* da categoria *Flow* diretamente abaixo do bloco verde para criarmos uma condição para que a mensagem apareça quando todas as frutas forem coletadas;

13. Na caixa de seleção do bloco `if`, vamos inserir o bloco com sinal de igual localizado na guia `Flow`;
14. Insira o atributo `Frutas` encontrado na categoria `Attributes` conforme a seguir para que, se as frutas chegarem a 0...:



Figura 7.17: Bloco de condição if no evento personalizado

15. Quando as frutas forem todas coletadas, faremos aparecer o *actor* `Stencyl Book Success`. Vamos inserir o bloco `create actor type at`, localizado na categoria `Scenes` para que ele surja a 160 pixels a partir do 0 da câmera, conforme visualizado a seguir:

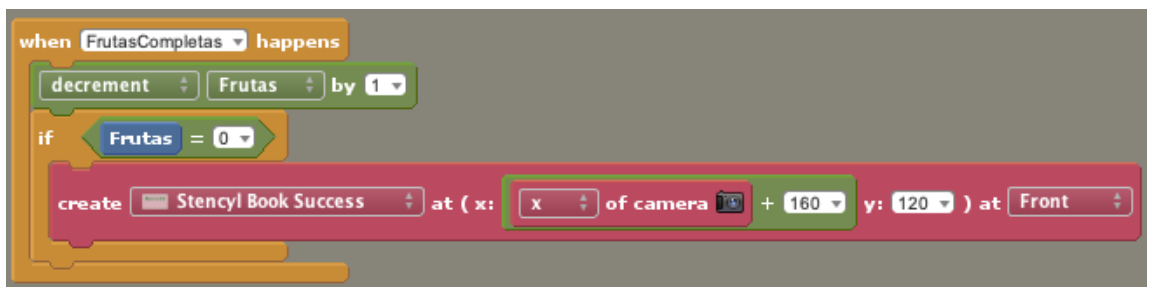


Figura 7.18: Bloco para exibição de actors na cena

16. Anexe este evento à *Cena01* clicando em *Attach to Scene* à direita da tela. Pronto! Quando as todas as frutas forem coletadas aparecerá a mensagem *Success*. Precisamos então inserir as duas instruções realizadas anteriormente no evento de colisão do nosso herói. Vamos finalizar o evento?
17. Abra o *actor behavior* *Colisões* e selecione o evento *Colidir Frutas* ;
18. Elimine todo o bloco *do after* da instrução e insira o bloco *if* para criarmos uma condição;
19. Insira o bloco de comparação de igualdade na caixa de seleção *if* ;
20. Anteriormente, utilizamos o bloco *set actor value* para configurar um valor falso para os *actors* do grupo *Frutas* , lembra? Vamos agora trazer para essa instrução aquele valor. Para isso, utilizaremos o bloco *get actor value* encontrado na categoria *Actors* e na guia *Properties*. Arraste-o para dentro do bloco *if* conforme a figura:



Figura 7.19: Bloco get actor correspondente a set actor

21. Nomeamos como *Coletadas* no *set actor* e atribuímos *false* para ele. Digite este nome, *Coletadas*, na caixa *value* e insira o bloco *false* da categoria *Flow* após o sinal de igual;
22. Arraste o bloco *actor of group* do bloco laranja para dentro do bloco *self* . Os valores serão atribuídos para o grupo *Frutas* :

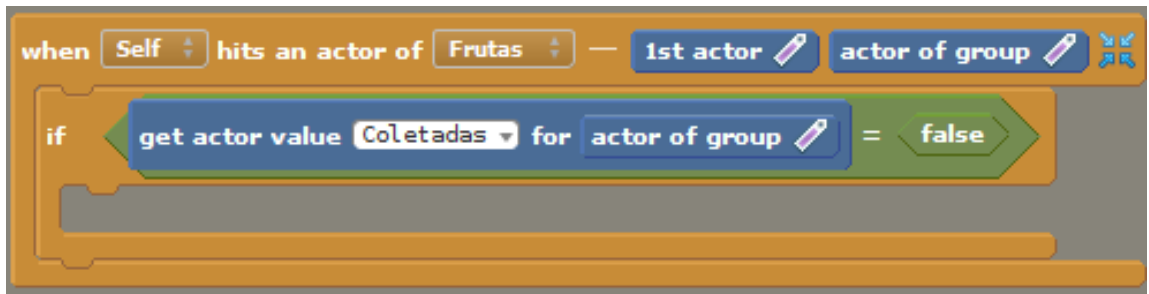


Figura 7.20: Configuração do bloco get actor value

23. Vamos inserir agora o bloco `set actor value` com o valor `true` para que este seja atribuído se todas as frutas forem coletadas. Veja:



Figura 7.21: Visualização do bloco set actor value

24. Se todas as frutas forem coletadas, queremos também que apareça o *actor* `Stencyl Book Success`. Personalizamos este evento anteriormente com o nome `FrutasCompletas`, e para engatilhá-lo nessa instrução precisaremos incluir o bloco do tipo *trigger* que já utilizamos anteriormente. Este bloco encontra-se na categoria `Behaviors`. Arraste-o para dentro do bloco `if` e insira o nome do evento e do *behavior* corretamente para que não ocorra erros na instrução:

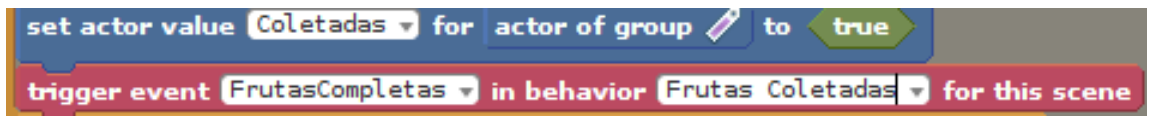


Figura 7.22: Bloco para o engatilhamento de eventos personalizados

25. Para completar, vamos eliminar todos os *actors* do grupo `Frutas`. Insira o bloco `kill actor` conforme a seguir:



Figura 7.23: Condição if para evento de colisão

Teste o jogo. Colete todas as frutas e confirme se a mensagem de sucesso aparece.

DICA:

Para eliminar os blocos que não forem mais utilizados da área de edição de eventos pressione as teclas `ctrl/cmd + k`.

Neste capítulo:

- aprendemos a reiniciar a cena;
- a dar mais chances ao *actor*;
- a desenvolver um *game over*, fim de jogo;
- a coletar itens.

No próximo capítulo, vamos criar novos níveis e fazer nosso personagem avançar por eles. Até lá!

CAPÍTULO 8

Criando mais níveis

Nosso primeiro nível está quase pronto, em breve criaremos outros níveis. Mas é importante concluir o primeiro nível pois os outros carregarão todos os eventos e instruções desenvolvidas para ele, bastando realizar as alterações necessárias e pontuais relacionadas a cada um.

Então, antes de criarmos os próximos níveis, vamos concluir o primeiro, configurando seu *background*, aprendendo a inserir uma imagem de segundo plano. E para compreendermos melhor sobre como trabalhar com o segundo plano da cena, conheceremos também, nos próximos tópicos, a importância do recurso das *Layers* (camadas).

8.1 Configurando o background

Inserir uma imagem de segundo plano é muito simples, basta compreender como funcionam as camadas (*layers*) no Stencyl. As *Layers* assemelham-se a papéis transparentes sobrepostos um sobre o outro, permitindo que "desenhemos" os personagens, *tiles* ou *backgrounds* em grupos diferentes. O *background*, por exemplo, deve ser incluído na camada mais abaixo, pois, caso contrário, cobrirá outras camadas que contêm os *tiles* e os personagens.

Antes de incluirmos o *background* na cena, precisaremos inserir a imagem no painel *Dashboard* na opção *Background*. Baixe o arquivo `jungle_background_1920x480` no link <https://github.com/scampelo/imgsBookStencyl> e siga as etapas. Este arquivo possui 1920 pixels de largura por 480 de altura:

1. No painel *Dashboard*, clique em *Backgrounds*;

2. Clique em `Create New` ;
3. Vamos digitar o nome *BackgroundCena01* e confirmar;
4. Nosso jogo ainda não possui nenhum *background*. Vamos inserir o arquivo `jungle_background_1920x480` que acabamos de baixar. Clique em `Click here to add a frame` à esquerda da tela;
5. Selecione **1x** na opção `Scale` para a imagem vir com a resolução original e, em seguida, `Choose Image` para localizar o arquivo;
6. Localize o arquivo `jungle_background_1920x480` e confirme;
7. A imagem aparecerá na janela `Add a frame` . Vamos confirmar mais uma vez em `Add` ;
8. Salve o jogo;
9. Pronto! A imagem foi incluída no Stencyl como *background*. À direita dessa tela, encontramos algumas opções de configuração:

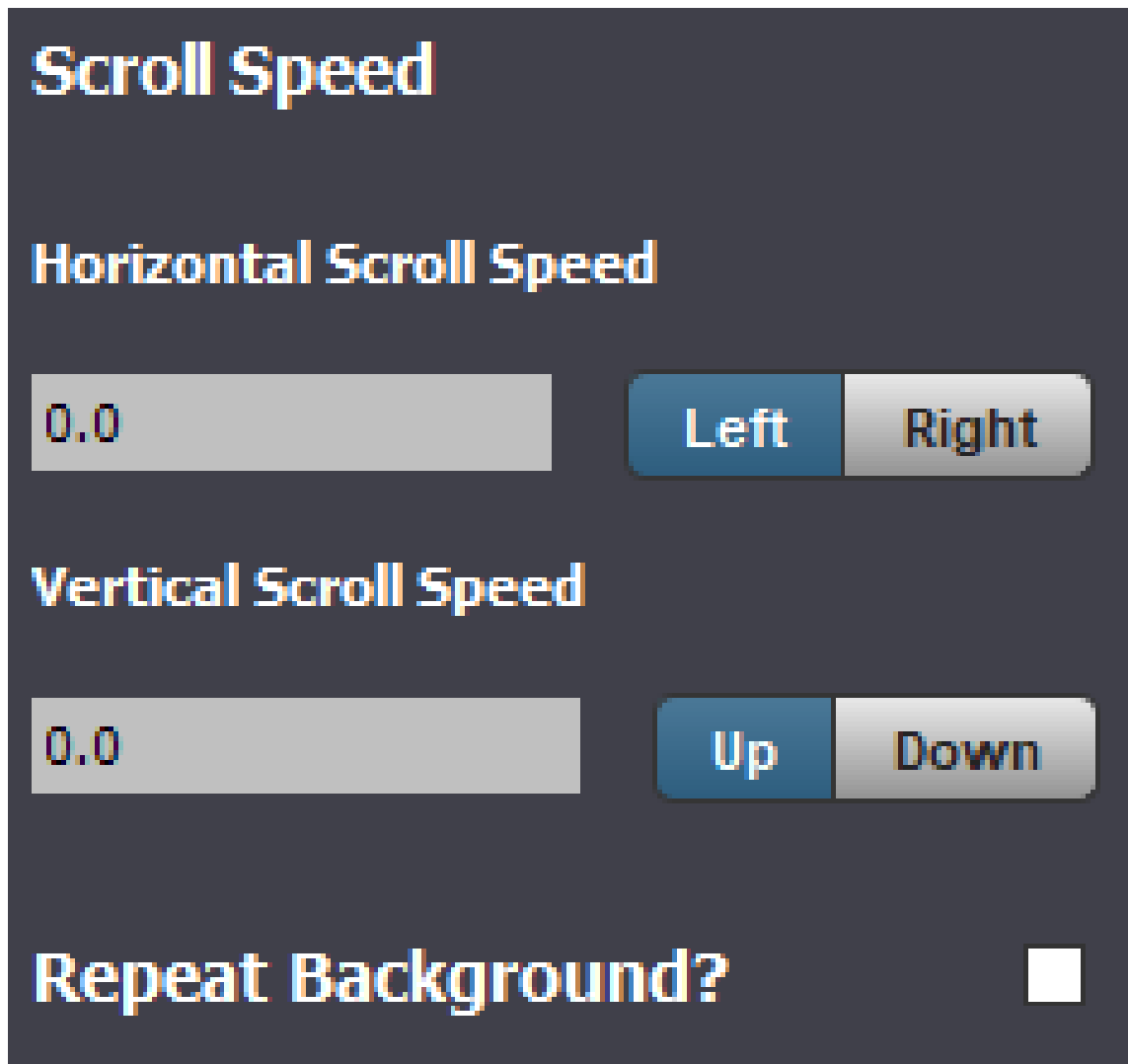


Figura 8.1: Área de configuração do rolagem background

- *Horizontal Scroll Speed*: rolagem do background na horizontal. Quanto maior o valor, mais rápida será a rolagem. O Stencyl permite que sejam incluídos vários *backgrounds* e que sejam ajustadas velocidades diferentes para cada um deles, criando o efeito que denominamos *parallax*. Como exemplo, se você está dentro de um carro em velocidade, os planos mais próximos rolam mais rapidamente, e os mais distantes, mais devagar.
- *Vertical Scroll Speed*: rolagem do background para cenários verticais;

- *Repeat Background*: utilize essa opção quando quiser que o *background*, tendo as dimensões menores do que a cena, seja repetido durante o desenrolar dela.
10. Como nosso *background* tem as dimensões da cena, não alteraremos os valores de configuração. Feche a janela do *background*, retorne ao *Dashboard* e abra a *Cena01*;
 11. No painel *Layers* (camadas), à direita da tela, temos apenas a camada *Layer 0* onde inserimos todos os *actors* e *tiles* que estão compondo nossa *Cena01*. Vamos inserir a camada do *background* para trazer à cena o arquivo de imagem incluído no *Stencyl*. Clique no sinal **+** do painel *Layer* e, em seguida, clique em *New Background Layer*;
 12. Selecione *BackgroundCena01* e clique em *Ok*;

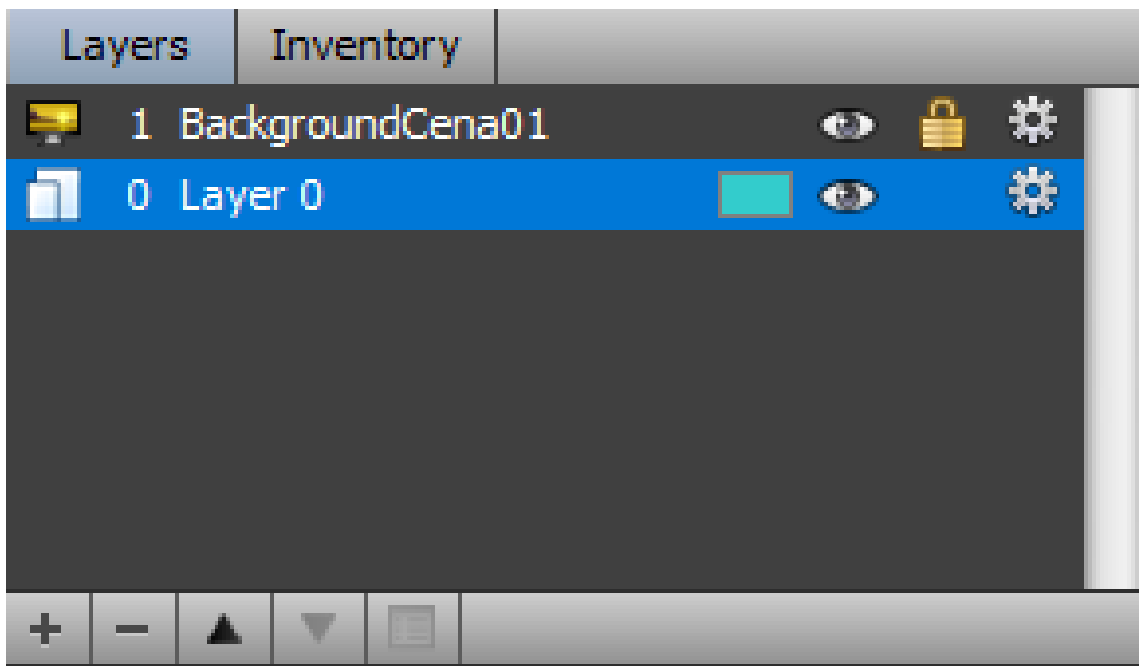


Figura 8.2: Painel de camadas com background

13. Repare que a camada *BackgroundCena01* está sobre a camada *Layer 0*. Vamos ordená-las corretamente, pois, quanto mais acima na lista, mais à frente estará na cena. Com a camada *Layer 0* selecionada, clique na seta preta abaixo do painel.

Dessa forma, esta camada, onde estão todos os *actors* e *tiles* da nossa cena, ficará sobre o *background*.

14. Teste o jogo.

DICAS:

- Para exibir ou ocultar uma camada, clique no olho correspondente;
- Para impedir alterações, clique no cadeado correspondente;
- Para remover uma camada, selecione-a e clique no sinal - do painel;
- Para renomear, configurar transparência ou configurar a rolagem, clique na engrenagem da camada correspondente.
- Preocupe-se sempre em importar as imagens como *background* com as dimensões já predefinidas, para que fiquem adequadas às cenas e aos dispositivos onde o jogo vai rodar.

8.2 Adicionando mais níveis ao jogo

Agora que concluímos nosso primeiro nível inserindo um novo *background*, vamos criar os novos níveis do jogo. Para que todas as instruções criadas para os *actors* e cena sejam levadas para os outros níveis, devemos tirar cópias dessa primeira cena.

Primeiramente, vamos aprender a renomear uma cena e colocar um nome mais adequado às novas missões:

1. No painel `Dashboard`, clique em `Scenes`;
2. Vamos clicar com o botão direito em *Cena01* e, em seguida, em `Rename`;
3. Vamos digitar *Level1*. É importante que o nome seja digitado corretamente, pois a contagem de níveis na instrução será

compreendida pela sequência numérica inserida após o nome. Pronto, renomeamos. Vamos agora tirar cópias desse primeiro nível:

4. Vamos clicar mais uma vez com o botão direito em *Level1* e, dessa vez em `Duplicate` ;
5. Renomeie o novo nível criado para *Level2*;
6. Repita o procedimento para criar mais um nível e renomeie-o para *Level3*.
7. Salve o jogo.

Utilizaremos apenas três níveis para dar sequência nas instruções para que nosso herói avance entre as fases do nosso jogo. Abra as cenas e clique, à direita da tela, em `Test Scene` . Repare que todas as instruções desenvolvidas para a primeira cena foram levadas para as outras duas. Para que nosso macaquinho avance entre os níveis precisaremos criar as novas instruções seguindo as próximas etapas.

Faça você mesmo!

Agora é a sua vez! Configure os níveis para que se diferenciem do primeiro. Afaste ou altere alguns *tiles*, reposicione *actors*, por exemplo. Lembre-se de que as instruções feitas anteriormente servirão para os níveis duplicados.

8.3 Avançando entre os níveis

Após duplicarmos a primeira cena, vamos iniciar o desenvolvimento dos *behaviors* para o avanço do nosso macaquinho entre os níveis. Primeiramente, semelhante ao *behavior* realizado para as vidas do nosso personagem principal, precisaremos criar um atributo para todo o jogo, um *game attribute*:

1. Vamos clicar no botão `Settings` na parte superior da tela;

2. À esquerda da janela, clique em `Attributes` ;
3. Clique em `Create New` e nomeie este atributo como *Level* e escolha o tipo `number` para que este valor sirva para a contagem dos níveis;
4. Em `Initial Value` , vamos inserir o valor **1.0** para que os níveis avancem nessa sequência;
5. Confirme e salve o jogo.

Com o atributo criado, vamos inserir a instrução para avançar entre os níveis. Nosso macaquinho avançará entre os níveis se conseguir coletar todas as frutas da cena, e esse evento já foi criado. Vamos abrir o *scene behavior* `Frutas Coletadas` e seguir as próximas etapas.

1. No *scene behavior* `Frutas Coletadas` , selecione o evento `Frutas Completas` ;
2. Vamos desabilitar a mensagem de êxito do bloco para posteriormente inseri-la no final dos níveis. Clique com o botão direito no bloco vermelho `create at` e selecione `Activate/Deactivate` ;
3. Vamos configurar o atributo criado anteriormente para que os níveis sejam contados de um em um, semelhante ao que fizemos no atributo `Vidas` . Na `Palette` , vamos clicar na categoria `Game Attributes` , na guia `Setters` e arrastar o bloco `set Level to` para dentro do bloco `if` ;
4. Vamos configurar o avanço de níveis. Leve o bloco de adição localizado na categoria `Numbers & Text` e solte-o dentro da caixa de seleção do bloco `set Level to` ;
5. No bloco posicionado antes do sinal de adição, insira o atributo `Level` localizado na categoria `Attributes` e na guia `Getters` :



Figura 8.3: Blocos da categoria atributos

6. Após o sinal de adição, vamos inserir o número **1** para que a contagem seja realizada nessa sequência;
7. Após configurar o avanço de níveis, vamos complementar com o avanço sequencial entre as cenas. Para isso, utilizaremos um bloco para a transição entre as cenas, chamado `switch to scene and crossfade for secs`, localizado na categoria `Scenes` e na guia `Game Flow`. Arraste-o e posicione-o abaixo do bloco `set Level to`:

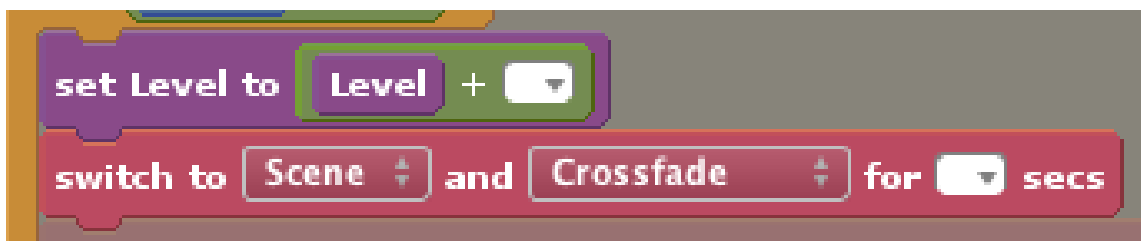


Figura 8.4: Bloco para transição entre cenas

8. Configure um efeito de transição em `crossfade` e um tempo para ela. Neste bloco, também já poderíamos escolher a próxima cena para avançar, clicando na opção `Scene` e `Choose Scene`. No entanto, precisamos que essa transição seja sequencial, ou seja, após completar o *Level/2*, nosso macaquinho avance também já para o *Level/3*, e assim por diante nos jogos em que existirem mais cenas. Para isso, utilizaremos um bloco para que possamos configurar essa instrução. Localize o bloco `scene with name` (cena com o nome) também em `Game Flow` e arraste-o para dentro caixa `Scene` do bloco vermelho;

9. Vamos inserir o bloco `anything as text` para converter valores numéricos no formato de texto. Na guia `Text` de `Numbers & Text`, arraste este bloco para dentro da caixa de seleção `scene with name`;
10. Arraste o bloco `text & text`, localizado na mesma guia anterior, para dentro do bloco `as text` para combinarmos duas informações de texto. Veja como estamos indo:

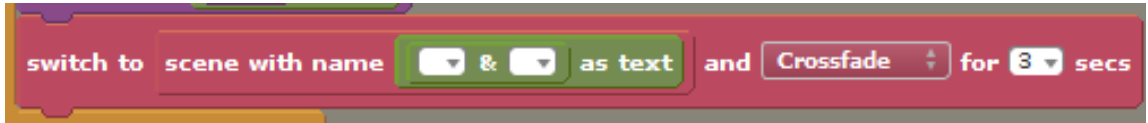


Figura 8.5: Blocos para utilização de textos

11. Para concluirmos a instrução, vamos inserir à esquerda no bloco verde o nome `Level` dado ao atributo criado anteriormente para a contagem numérica. Veja como ficou:

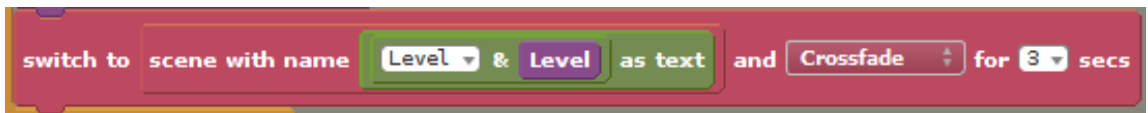


Figura 8.6: Instrução para transição entre cenas com atributo numérico

12. Salve e teste o jogo.

Veja se os níveis estão avançando automaticamente sempre que nosso macaquinho coleta todas as frutas. Lembre-se de que o evento só está funcionando porque ele já está engatilhado no evento `Coletar Frutas`.

Neste capítulo:

- aprendemos a criar novos níveis para o jogo;
- a configurar um *background* para as cenas;
- a avançar entre os níveis.

No próximo capítulo, vamos finalizar nosso jogo para, depois, aprendermos as formas de divulgação e publicação.

CAPÍTULO 9

Finalizando o jogo

Neste capítulo, vamos finalizar nosso jogo. Criaremos uma tela inicial, aprenderemos a inserir botões e também finalizaremos com a inserção de efeitos de som e trilha sonora.

Primeiramente, vamos já baixar alguns *actors* lá no StencylForge :
Stencyl Book Play Button , Stencyl Book Start Button , Stencyl Book Menu Button e Stencyl Book Title Banner . Não esqueça de desativar a movimentação desses *actors* em Cannot Move na guia Physics .

9.1 Funções de botões aos actors

Neste tópico vamos configurar algumas funções de botões a *actors* para o acesso a outras cenas. Primeiramente, vamos criar duas novas cenas seguindo as etapas a seguir:

1. Crie duas novas cenas para o jogo com os nomes *Level4* e *Menu Inicial*. Insira nas cenas os *actors* baixados. Veja os exemplos:



Figura 9.1: Cena inicial

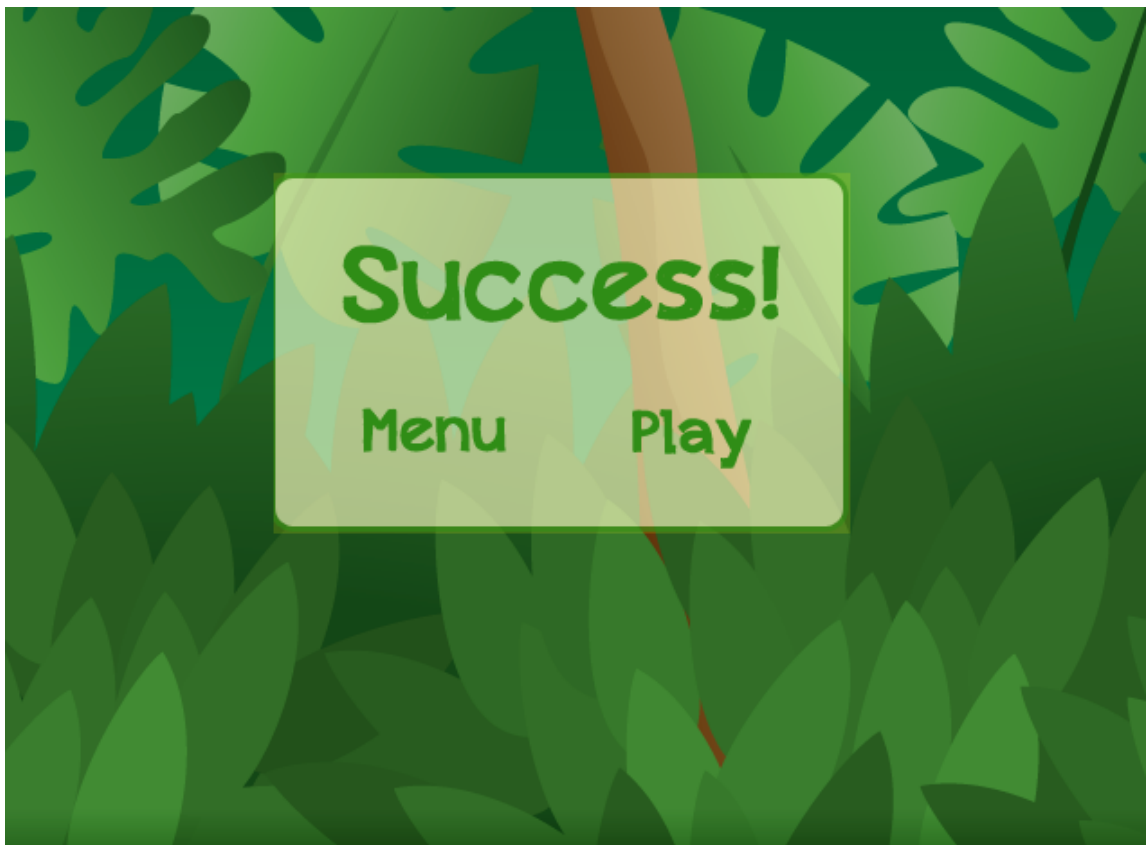


Figura 9.2: Cena final

2. Vamos tornar o menu inicial a primeira tela do jogo quando ele for iniciado. Selecione a cena de mesmo nome em `Scenes`, (não abra, apenas selecione) e clique no botão `Mark as Starting Scene` no canto inferior direito da tela:



Figura 9.3: Opção para marcação da tela inicial do jogo

3. Repare que a cena inicial agora tem uma estrela no seu canto superior esquerdo. Agora vamos configurar as funções de botões para os *actors*. Primeiramente, vamos configurar o

acesso à tela inicial. Abra o *actor* Stencyl Book Menu Button em Actor Types ;

4. Criaremos um evento para que, quando o mouse for pressionado sobre este *actor*, apareça o menu inicial do jogo, ou seja, a cena que acabamos de criar. Em Events , adicione um novo evento em Input e, depois, clique em On Actor ;
5. Renomeie-o para *ClickMenuInicial*;
6. O bloco when the mouse is released on self está configurado para quando o mouse for pressionado e liberado do *actor* uma ação ocorrer. Vamos incluir o bloco para a transição entre cenas. Na categoria Scenes , em Game Flow , arraste o bloco switch to scene and crossfade for secs para dentro do bloco laranja;
7. Configure o bloco conforme a seguir para que a cena Menu Inicial apareça após o clique sobre o *actor*:



Figura 9.4: Bloco para transição entre cenas

8. Salve e teste a cena *Level4* e confirme se a transição funcionou.

Faça você mesmo!

Abra o *actor* Stencyl Book Menu Button e crie um evento para que, ao clicar sobre ele, o jogo se inicie, ou seja, a cena *Level1* apareça. Configure também o Stencyl Book Play Button com a mesma instrução.



Figura 9.5: Bloco para transição entre cenas

Inserindo actors (botões) com eventos

No tópico anterior, antes de configurarmos os *actors* com funções de botão, criamos as cenas e depois desenvolvemos as instruções para as transições entre elas. Dessa vez, vamos inserir os *actors*/botões na cena por meio de eventos. Utilizamos este recurso no tópico *Inserindo novos actors na cena* quando fizemos as estátuas caírem na tela. Dessa vez, vamos inserir um *actor* quando a mensagem de *Game Over* aparecer, para que possamos voltar a jogar novamente.

1. Vamos abrir o *actor behavior* vida e clicar no evento *Mostrar Anjo* ;
2. Criamos a instrução *Mostrar Anjo* para que o macaco em forma de anjo surja quando ele colidir com os inimigos ou quando a contagem regressiva acabar; e se as 02 chances terminarem, surgirá o *8actor Game Over* . *Vamos inserir agora, junto a este último actor, o actor Stencyl Book Play Button para que possamos ter a opção de reiniciar o jogo. Para não inserirmos o mesmo bloco de criação de actors* na cena novamente, vamos copiá-lo. Clique sobre o bloco create Stencyl Book Game Over Banner mantendo pressionada a tecla alt/option e arraste-o conforme imagem:*

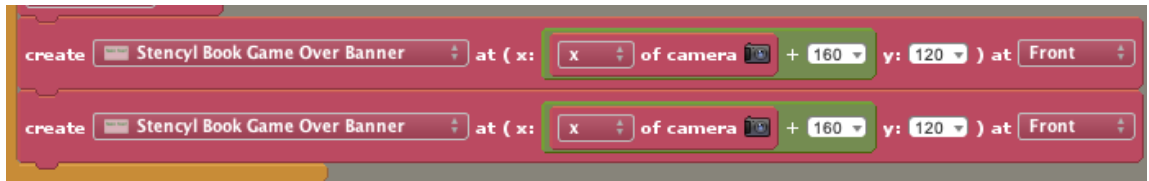


Figura 9.6: Blocos para criação de actors nas cenas

3. Insira o *actor* Stencyl Book Play Button no campo create e configure seu posicionamento com o valor de **270** no campo verde para a coordenada x e **230** para o y. Ele surgirá abaixo do *actor* Game Over :

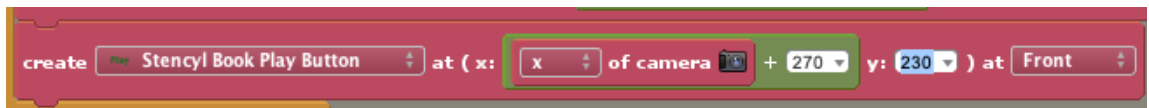


Figura 9.7: Instrução para criação e posicionamento do actor na cena

Antes dos blocos de criação dos *actors*, configuramos para o jogo pausar. Para que o *actor* Stencyl Book Play Button continue ativo mesmo com o jogo pausado, precisamos habilitar a opção Can be Paused? como No . Vamos seguir as próximas etapas para habilitar esta opção:

4. Vamos voltar ao Dashboard e abrir o *actor* Stencyl Book Play Button ;
5. Na guia Physics , em Advanced , selecione a opção No em Can be Paused? . Dessa forma o *actor* continuará ativo mesmo com o jogo pausado:

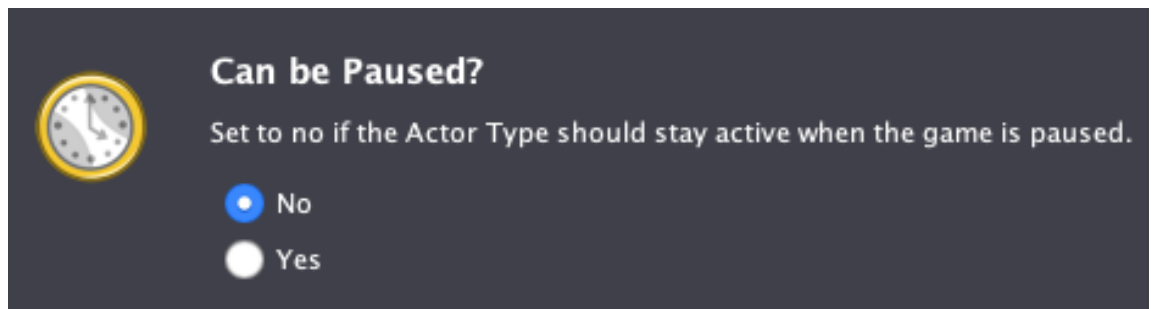


Figura 9.8: Opção para manter o actor ativo com o jogo pausado

6. Vamos configurar o *actor* `Stencyl Book Play Button` para que, quando clicarmos sobre ele, a pausa do jogo seja cancelada e seja assumida a configuração das chances novamente para o *actor*. Clique na guia `Events` deste *actor*;
7. Na `Pallette`, clique na categoria `Scenes` e na guia `Game Flow`;
8. Arraste o bloco `pause game` e solte-o acima do bloco `switch to`. Selecione `unpause game` para cancelar a pausa;
9. Para concluirmos o evento, vamos inserir um bloco para que nosso *actor* volte a ter 3 chances no jogo. Clique em `Game Attributes` e na guia `Setters` arraste o bloco `Set Vidas to` para dentro do bloco laranja. Veja a seguir:



Figura 9.9: Bloco para configuração das chances do actor

10. Salve o jogo.

Assim que o nosso macaquinho perder as 3 chances, a mensagem *Game Over* surgirá com a opção *Play* para retornarmos ao jogo.

9.2 Efeitos de som e trilha sonora

Para concluirmos nosso jogo, está faltando apenas inserirmos os efeitos de som e uma trilha sonora. Quando pensamos em som no jogo, estamos pensando em melhorar a experiência e aumentar a imersão de quem joga, não é verdade? São dois os tipos de sons que podemos utilizar no Stencyl:

Sound Effects: utilizados para sons curtos, pois consomem mais memória.

Music: utilizados para sons maiores (trilha sonora) e são rodados via *streaming* (internet).

Sons como efeitos são áudios incluídos durante os comportamentos ou ações dos *actors* no jogo. Podemos inserir som durante a movimentação dos *actors*, durante sua queda, ou quando ele colide com algum outro *actor*, por exemplo. Trilha sonora é aquela música que permanece de fundo, uma música que segue junto durante a jogabilidade, um tema para o ambiente do jogo.

O método de utilização de sons no Stencyl assemelha-se à utilização de todos os outros elementos que utilizamos durante o projeto de nosso jogo. Precisamos primeiramente importá-lo para o software para depois incluí-lo nas instruções. Os formatos de arquivos de som que o Stencyl suporta são MP3 e OGG, pois são arquivos leves e adequados às suas plataformas de saída.

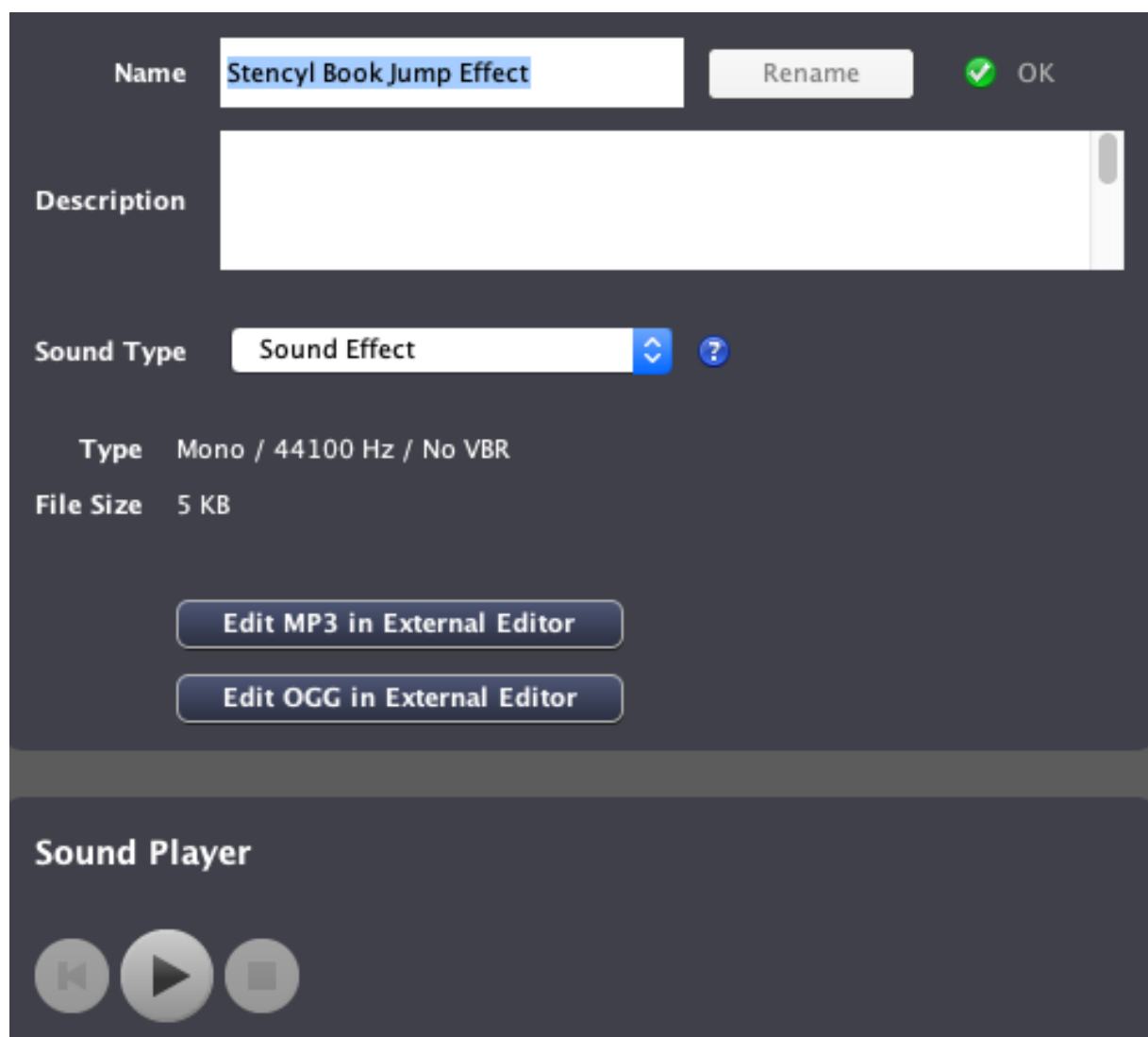
| Platform | Accepted Format |
|----------|-----------------|
| Flash | MP3 |
| HTML5 | OGG |
| Windows | OGG |
| Mac | OGG |
| Linux | OGG |
| iOS | OGG |
| Android | OGG |

Figura 9.10: Formatos de som e suas plataformas

Além disso, arquivos de áudio com taxas de amostragem de 44.1 KHz e *bit rate* (taxa de bits) constante são melhores para utilização no software.

Para continuarmos, vamos baixar alguns arquivos de som no StencylForge . Localize os arquivos de som Stencyl Book Jump Effect , Stencyl Book Pickup Effect e Stencyl Book Sound Track e baixe-os. Eles aparecerão na opção Sound no Dashboard .

No Dashboard , em Sound , abra um dos arquivos, que você verá a tela a seguir. Nela você pode visualizar a configuração do arquivo, seu tipo (para efeito ou trilha sonora) e testá-lo:



The image shows a dark-themed user interface for managing sound files. At the top, there is a 'Name' field containing 'Stencyl Book Jump Effect', a 'Rename' button, and a green checkmark with 'OK'. Below this is a 'Description' field. The 'Sound Type' is set to 'Sound Effect' with a dropdown arrow and a help icon. Further down, the 'Type' is 'Mono / 44100 Hz / No VBR' and the 'File Size' is '5 KB'. There are two buttons: 'Edit MP3 in External Editor' and 'Edit OGG in External Editor'. At the bottom, there is a 'Sound Player' section with three circular buttons: a previous button, a play button, and a stop button.

Name **Stencyl Book Jump Effect** Rename ✓ OK

Description

Sound Type **Sound Effect** ?

Type Mono / 44100 Hz / No VBR

File Size 5 KB

Edit MP3 in External Editor

Edit OGG in External Editor

Sound Player

⏮ ▶ ⏹

Figura 9.11: Janela referente ao arquivo de som

DICA:

Para importar um arquivo de som, basta clicar em **Sounds** no **Dashboard**, em seguida em **Click here to create a new Sound**, inserir um nome para ele e importar o arquivo clicando em **Import MP3 / Import OGG**. As opções vistas são as mesmas para o arquivo importado. No site <https://opengameart.org> você pode baixar gratuitamente e livremente vários arquivos de áudio para serem utilizados no Stencyl.

Criando *actors behaviors* para efeitos de som

Os efeitos de som podem ser inseridos diretamente nos eventos realizados para os *actors* (*actor behaviors*) ou para as cenas (*scene behaviors*). Vamos iniciar criando um evento para os pulos do nosso macaquinho:

1. Vamos criar um novo *actor behavior* chamado *Efeitos de Som*;
2. Adicione um evento em **+ Add Event**, **Input**, **Keyboard** (para teclados) e renomeie-o para *Som Pulo*;
3. No bloco laranja, vamos escolher a tecla correspondente ao pulo (*up*) e a ação para pressioná-la. Veja:



Figura 9.12: Bloco do evento Input Keyboard

4. Agora vamos incluir o áudio inserido no **Stencyl** anteriormente. Na **Pallette**, vamos clicar em **Sounds** e arrastar o bloco **Play Sounds** para dentro do bloco laranja;

5. Clique na caixa de seleção `Sound` e em seguida selecione `Choose Sound` ;
6. Selecione o arquivo `Stencyl Book Sound Effect` e clique em OK;

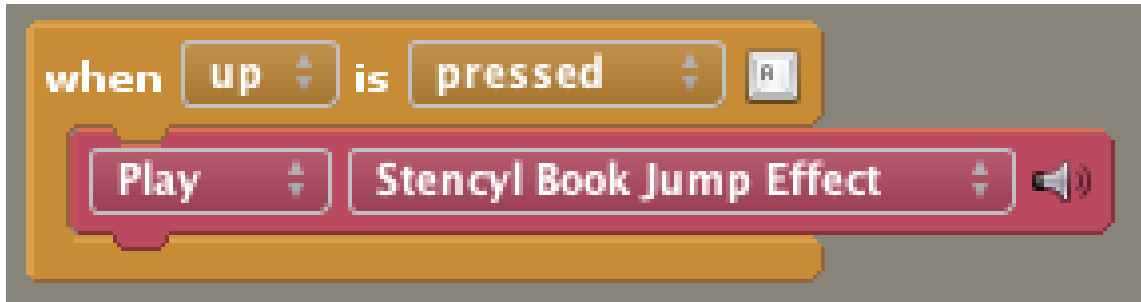


Figura 9.13: Evento para efeito de som

7. Salve e teste o jogo.

Pule com o macaquinho e curta o efeito de som!

Inserindo efeitos de som nos actors

Os eventos para inserção de som são muito simples, bastando definirmos onde os sons serão aplicados. No tópico anterior, inserimos o bloco de efeitos de som para a movimentação do *actor* pelo teclado. Vamos agora inserir sons no momento do clique do mouse diretamente sobre o *actor*. Veja o exemplo:

1. Abra o *actor* `Stencyl Book Start Button` no Dashboard . Vamos inserir um efeito de som sobre este *actor* quando clicarmos nele para acessar outra cena;
2. Em `+ Add Event` , desça até `Input` e clique em `On Actor` ;
3. Vamos renomear este evento para *SomClick*;
4. Na `Pallette` , na categoria `Sounds` , arraste o bloco `Play Sound` para dentro do bloco laranja e escolha o som que desejar clicando em `Sound` , `Choose Sound` ;
5. Salve e teste o jogo.

Clique sobre o *actor* `Start` e ouça o som escolhido.

Faça você mesmo!

Insira sons nos outros *actors* Stencyl Book Play Button e Menu Button .

Inserindo efeitos de som em eventos

Já inserimos um efeito de som durante a movimentação do *actor* e também diretamente nele. Dessa vez, vamos inserir um efeito de som no momento em que nosso protagonista coletar as frutas, dentro de um evento. Precisaremos, então, acessar o *actor behavior* de colisão:

1. No Dashboard , vamos abrir o *actor behavior* Colisões ;
2. Após selecionarmos o evento Colidir Frutas , vamos encaixar o bloco Play Sound acima do bloco if para que ele não entre dentro da condição:

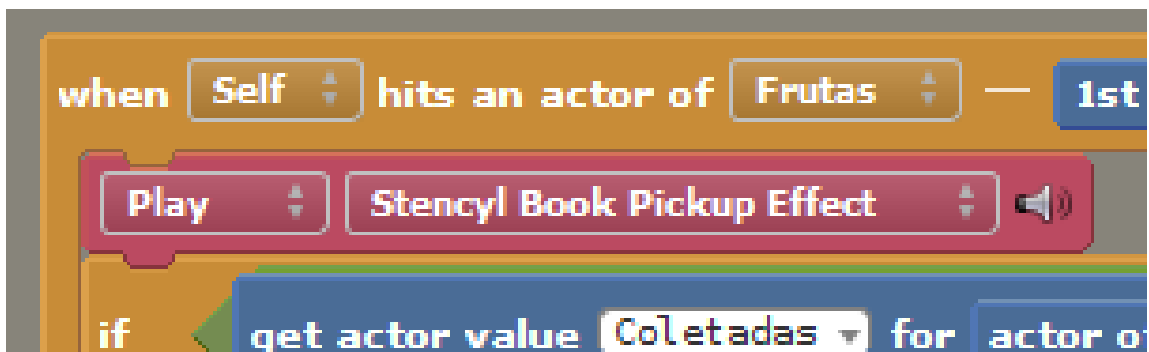


Figura 9.14: Bloco de som em evento de colisão

3. Selecione o arquivo de áudio Stencyl Book Pickup Effect e salve o jogo.

Colete algumas frutas para ouvir o som.

Faça você mesmo!

Tente incluir um efeito de som nas colisões com os inimigos. Não esqueça de incluí-los primeiramente no Dashboard .

Inserindo trilha sonora

Para concluirmos este módulo, vamos inserir um arquivo de som para servir como trilha sonora no jogo. Para isso, aplicaremos o som diretamente na cena, no momento de seu carregamento. Vamo utilizar como exemplo a cena `Level1` :

1. Abra a cena `Level1` em `Scenes` no `Dashboard` ;
2. Vamos criar um novo evento para quando a cena for carregada. Em `+Add Event` , clique em `Basics` e `When Creating` ;
3. Renomeie este evento para *Trilha01* ;
4. Dessa vez, utilizaremos o bloco de som `Play Sound on channel` para que não haja conflito entre os sons do jogo. Utilizando este bloco, você pode configurar os sons separadamente, alterando seus volumes e interrompendo-os, por exemplo, evitando que um interfira no outro. Em `Sounds` , na `Pallette` , arraste o bloco `Play Sound on channel` para dentro do bloco laranja;
5. Escolha o arquivo de som `Stencyl Book Sound Track` em `Sound` e digite `1` no campo editável para definir o canal de som;
6. Em `Play` , selecione `Loop` , para que o som se repita sempre que chegar ao final.

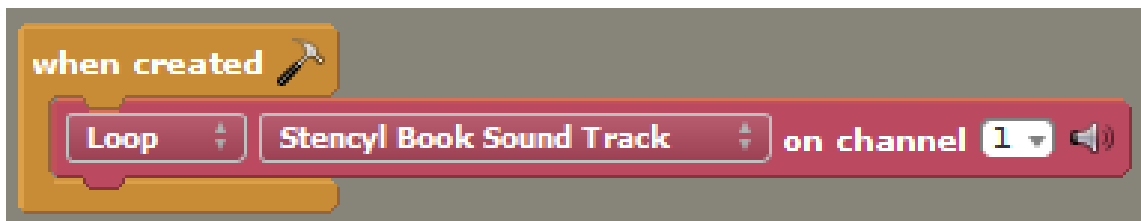


Figura 9.15: Bloco para inserção de trilha sonora

7. Salve e teste o jogo.

DICAS:

- Caso queira reduzir o volume da trilha sonora, inclua o bloco `set volume to` encontrado em `Sounds` e reduza a porcentagem do som. Posicione-o abaixo do bloco `Loop Sound` ;
- Para interromper o som na próxima cena, utilize o bloco `stop sound on channel` e especifique o mesmo canal utilizado na cena anterior.

Neste capítulo:

- configuramos *actors* como botões;
- inserimos *actors* nas cenas por eventos;
- trabalhamos com efeitos de som e trilha sonora.

No próximo e último capítulo, veremos como divulgar e distribuir o jogo nas plataformas.

CAPÍTULO 10

Testando e publicando o jogo

Após a conclusão do jogo, e antes de publicá-lo, é importante que o testemos para visualizar se todos os elementos inseridos no jogo estão funcionando adequadamente. Várias são as plataformas de destino em que podemos divulgá-lo, como vimos no início do livro: dispositivos móveis (iOS, Android), desktops (Linux, Windows, Mac OS) e em *web browsers* (Flash e HTML5). Neste capítulo, aprenderemos a testá-lo e também a complementá-lo com recursos para posterior divulgação.

Para testar o jogo, podemos inserir o dispositivo móvel por meio do cabo de conexão ao desktop com sistema operacional Windows ou Mac OS e visualizar diretamente no dispositivo (o sistema operacional Linux só permite testar o jogo no Android), mas para isso temos que primeiro nos registrar nos sites das respectivas empresas.

Com relação à sua publicação, o jogo pode ser disponibilizado nas lojas virtuais Google Store (Android) e App Store (iOS), com custo nas duas plataformas. Quando for testar seu jogo no iOS ou no Android, será necessário instalar alguns complementos, como a linguagem Xcode (Apple) e o Android SDK.

10.1 Inserindo ícone no jogo

Antes de aprendermos a divulgar e distribuir o jogo, é importante identificá-lo por meio de uma imagem (logo) para que seja reconhecido facilmente na plataforma. O logo é uma imagem em **.png**, leve, com dimensões que variam entre 1024 por 1024 pixels para que sirva como uma identificação estética nas plataformas

onde será visualizado. Essa imagem será redimensionada conforme a plataforma de saída. Inserir este logo é muito simples. Vamos baixar duas imagens: `logo_game` e `icon game` no link <https://github.com/scampelo/imgsBookStencyl> e seguir as etapas:

1. Vamos abrir as configurações gerais do nosso jogo. Clique em `Settings` na barra superior do Stencyl;
2. No primeiro item também chamado `Settings`, clique na guia `Main`:

The screenshot shows the 'Main' settings window in Stencyl. It has a dark gray background. At the top, there's a 'Name' field with the text 'Jogo do Macaco'. Below it, a smaller text says 'Only use letters, numbers and spaces.' Below that is a 'Description' field. Further down, there's a 'Game Logo' section with a red warning triangle icon and a 'Select Image' button. At the bottom, there's an 'Icon' section with a red square icon containing the word 'Icon' and another 'Select Image' button.

Figura 10.1: Janela para inclusão de ícone e logo no jogo

3. Encontramos opções para tirar uma cópia do jogo (`save game` as ao final da linha `Name`), fazer uma descrição sobre o jogo (`Description`), inserir um logo no jogo no Stencyl (`Game Logo`) e inserir um ícone no jogo para as plataformas de saída (`Icon`). Clique em `Select Image` nas duas últimas opções, localize as imagens baixadas anteriormente e as inclua como logo e ícone.

4. Confirme em `OK`.

Salve o jogo. Visualizaremos o logo apenas quando fecharmos o jogo no Stencyl. O ícone será visualizado na plataforma.



Figura 10.2: Logo e ícone do jogo

10.2 Configurando tela e Splash Screens

Precisamos também configurar como sua tela se adequará ao dispositivo e suas imagens de carregamento. Para isso, vamos acessar as configurações do jogo continuando na opção `Settings` :

1. Em `Settings`, vamos clicar no painel à esquerda em `Mobile` para configurar nosso jogo para os dispositivos móveis;
2. Clique em `General` :

The image shows a dark-themed settings window with the following fields and descriptions:

- App Name:** A text input field containing "My Game". Below it, the text reads: "The name that appears under the game's icon."
- App ID:** A text input field containing "com.yourname.gamename". Below it, the text reads: "A unique, internal name for the App Store / Google Play. (ex: com.stencyl.balloons)"
- Version:** A text input field containing "1.0". Below it, the text reads: "User-visible version such as '1.0' or '2.2'"
- Version Code (iOS):** A numeric input field with a spinner, containing the value "1". Below it, the text reads: "For internal versioning on the App Store. Must be an integer (1,2,3)."
- Version Code (Android):** A numeric input field with a spinner, containing the value "1". Below it, the text reads: "For internal versioning on Google Play. Must be an integer (1,2,3)."

Figura 10.3: Janela de configurações gerais para dispositivos móveis

- **App Name:** aqui você digitará o nome do seu jogo que aparecerá no dispositivo;
 - **App ID:** código de identificação que você recebeu na plataforma Apple;
 - **Version:** versão do jogo;
 - **Version Code:** número de controle da versão para as plataformas.
3. Vamos clicar na guia **Display** para configurarmos resolução e escalonamento da tela:

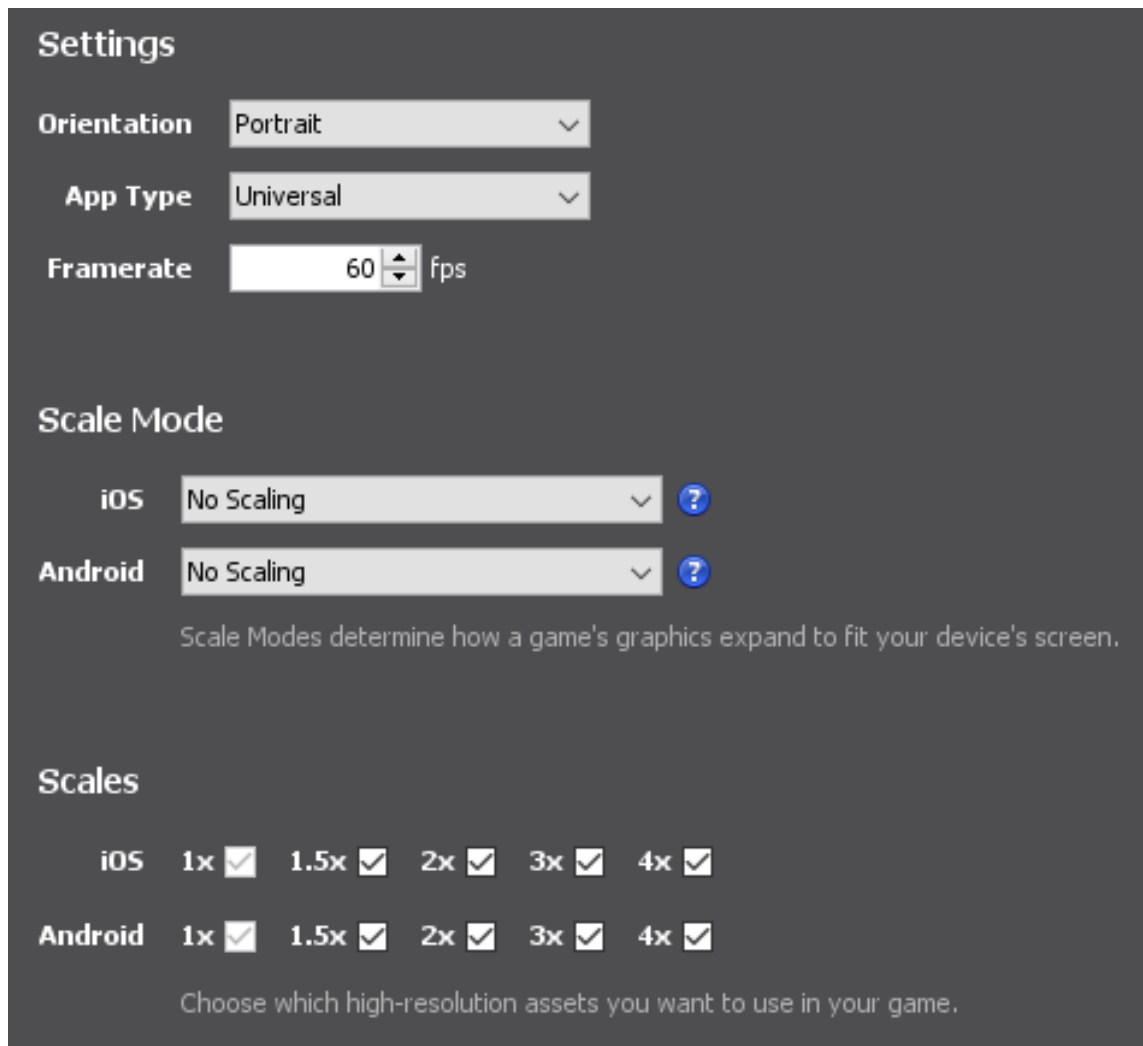


Figura 10.4: Configurações de tela e resolução

- **Settings:**
- **Orientation:** orientação horizontal ou vertical para o jogo;
- **App Type:** definição se o jogo rodará em *smartphones*, *tablets*, ou em ambos;
- **Framerate:** quadros por segundo de processamento do jogo;
- **Scale Mode:** opções de escalonamento da tela para as duas plataformas;
- **Scales:** definição das resoluções que poderão ser utilizadas no jogo, dependendo do dispositivo.

DICA:

Todas as opções desta janela também são encontradas para a configuração dos jogos para *Web* (Flash e HTML) e para *Desktop* (sistemas operacionais)

4. Para concluir, vamos clicar na opção `Splash Screens` . Aqui configuraremos as imagens de carregamento do jogo:



Figura 10.5: Configuração da tela de carregamento

- *Scale Mode*: modo de escalonamento da imagem que aparecerá durante o carregamento;
- *Background*: cor da tela do carregamento;
- *Splash*: imagens que aparecerão durante o carregamento.

10.3 Testando o jogo

Pronto, com isso, podemos testá-lo. Durante o livro, testamos no *Flash* ou em *HTML5*, agora vamos aprender os procedimentos para testes nas plataformas *mobile*. Veremos a seguir que para testar nos dispositivos *Android* e *iOS* faz-se necessário baixar as respectivas plataformas de desenvolvimento e, se necessário, algumas linguagens complementares.

Android

Para testar seu jogo em um dispositivo *Android*, será necessário primeiramente a instalação de alguns complementos. Selecione a opção *Android* em *Platform*, no canto superior direito do *Stencyl* e clique em *Test Game*. Aparecerá a mensagem solicitando a instalação do complemento *Android SDK* (ferramenta para o desenvolvimento de *apps* para *Android*). Confirme, que a próxima tela aparecerá:

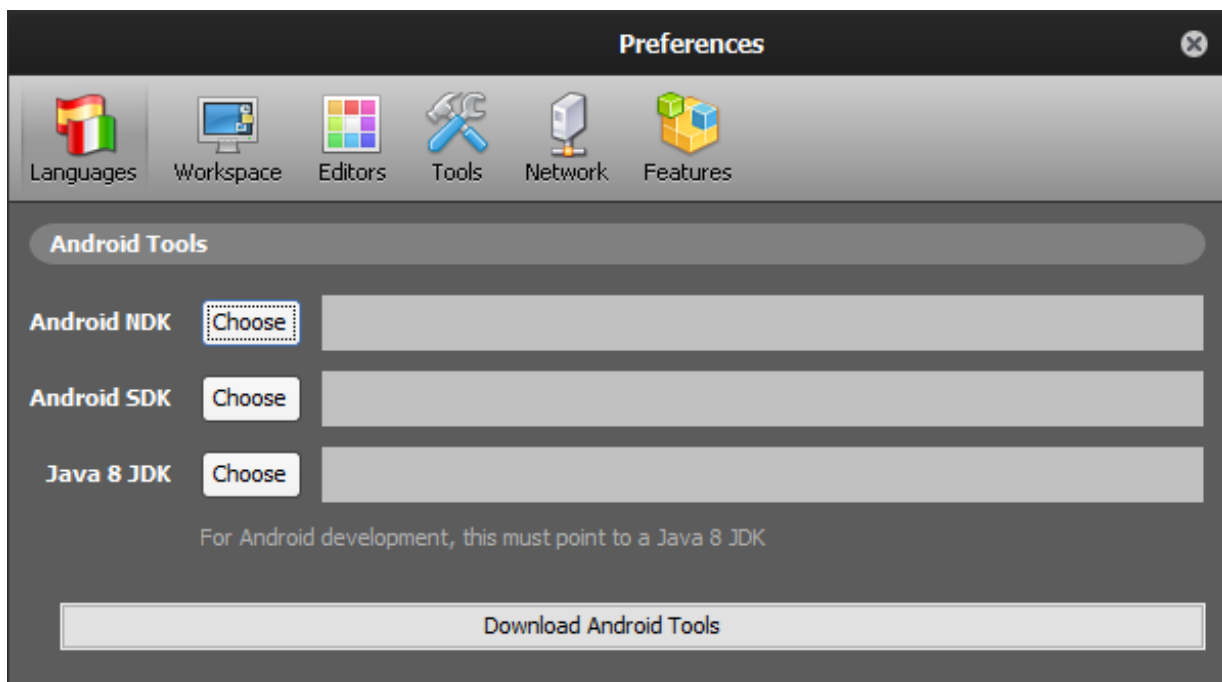


Figura 10.6: Tela para atualização dos complementos Android

Clique na opção *Download Android Tools* para a instalação dos complementos. Baixe também a linguagem *Java 8 JDK*, ambiente

para o desenvolvimento de aplicativos desenvolvida pela empresa americana Oracle.

iOS

Para testar o jogo em dispositivos iOS, o procedimento é parecido. Após clicar em `Test Game` e selecionar `iOS Device`, será solicitado que a linguagem de desenvolvimento *Xcode para Apple* seja instalada, mas para isso será necessária a criação de uma conta de desenvolvedor no site da Apple. É o que veremos no próximo tópico.

10.4 Publicando o jogo

Após os testes, vamos aprender agora a publicar o jogo nas plataformas *Google Play* e *App Store*, para que os usuários tenham acesso ao seu projeto. As duas plataformas necessitam da criação de uma conta de desenvolvedor para que sejam instalados alguns recursos no Stencyl para a publicação.

App Store

Primeiramente, é necessário criar uma conta de desenvolvedor no site da Apple:

1. Acesse <https://developer.apple.com>;
2. Clique em `Account` para acessar ou criar sua nova conta de desenvolvedor Apple;
3. No site, você deverá se inscrever no `Apple Developer Program` (Programa de Desenvolvedores Apple) para ter acesso à distribuição, compartilhamento e gerenciamento de seus *apps* e jogos na *App Store*. Neste site você obterá seus certificados e criará seus identificadores (IDs) e perfis de autorização para a publicação dos jogos;

4. Após a inscrição, você terá acesso aos arquivos complementares que deverão ser baixados, instalados e configurados no Stencyl na opção **Settings** e na guia **Certificates (iOS)** em **Mobile**. Veja a janela:

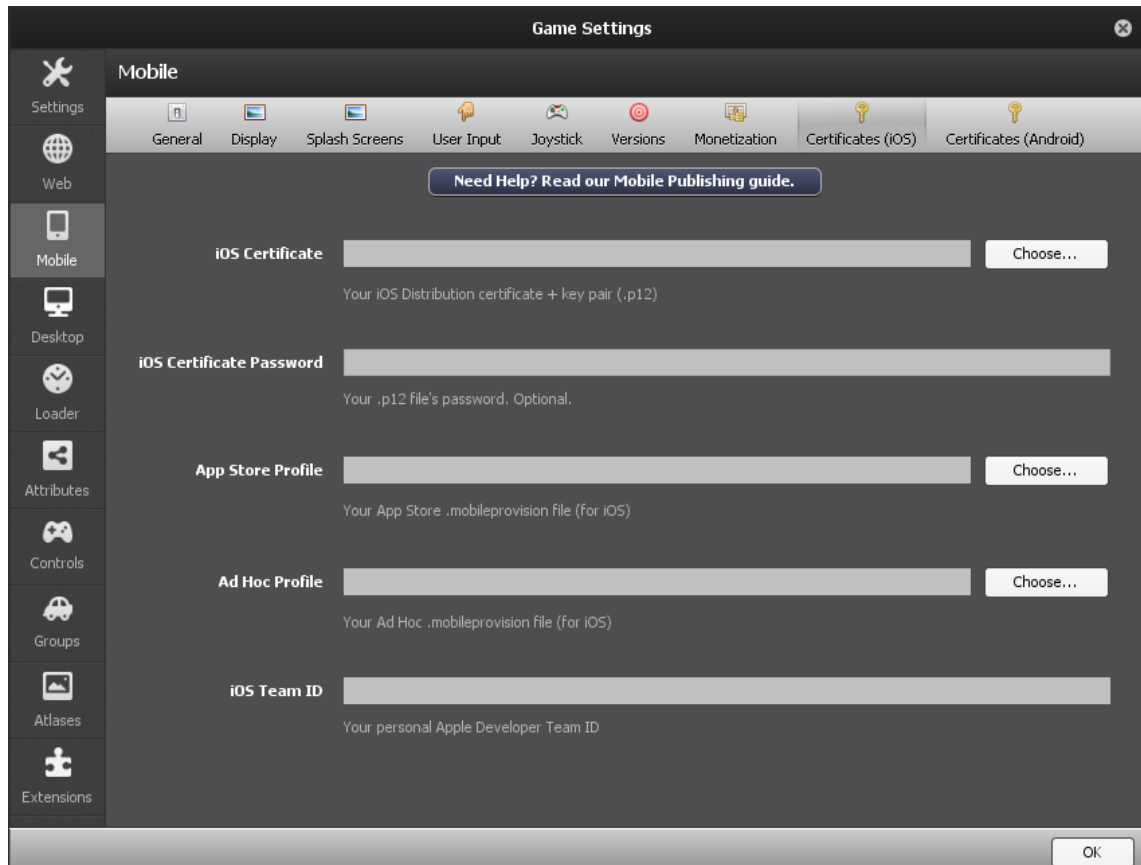


Figura 10.7: Tela para inclusão dos certificados Apple

- **iOS Certificates**: certificados para o desenvolvimento, testes, compartilhamento e distribuição do jogo na *App Store*;
- **iOS Certificate Password**: após a obtenção dos certificados, é necessário gerar um arquivo no formato **.p12** para associá-lo no Stencyl. Na geração deste arquivo, solicita-se a criação de uma senha que pode ser incluída neste campo opcionalmente;
- **App Store Profile**: código de identificação (ID) criado no site da Apple;

- *Ad Hoc Profile*: código do perfil de autorização criado no site da Apple de compartilhamento para testes beta;
- *iOS Team ID*: o identificador para membros do *Apple Developer Program*.

5. Veja o exemplo:

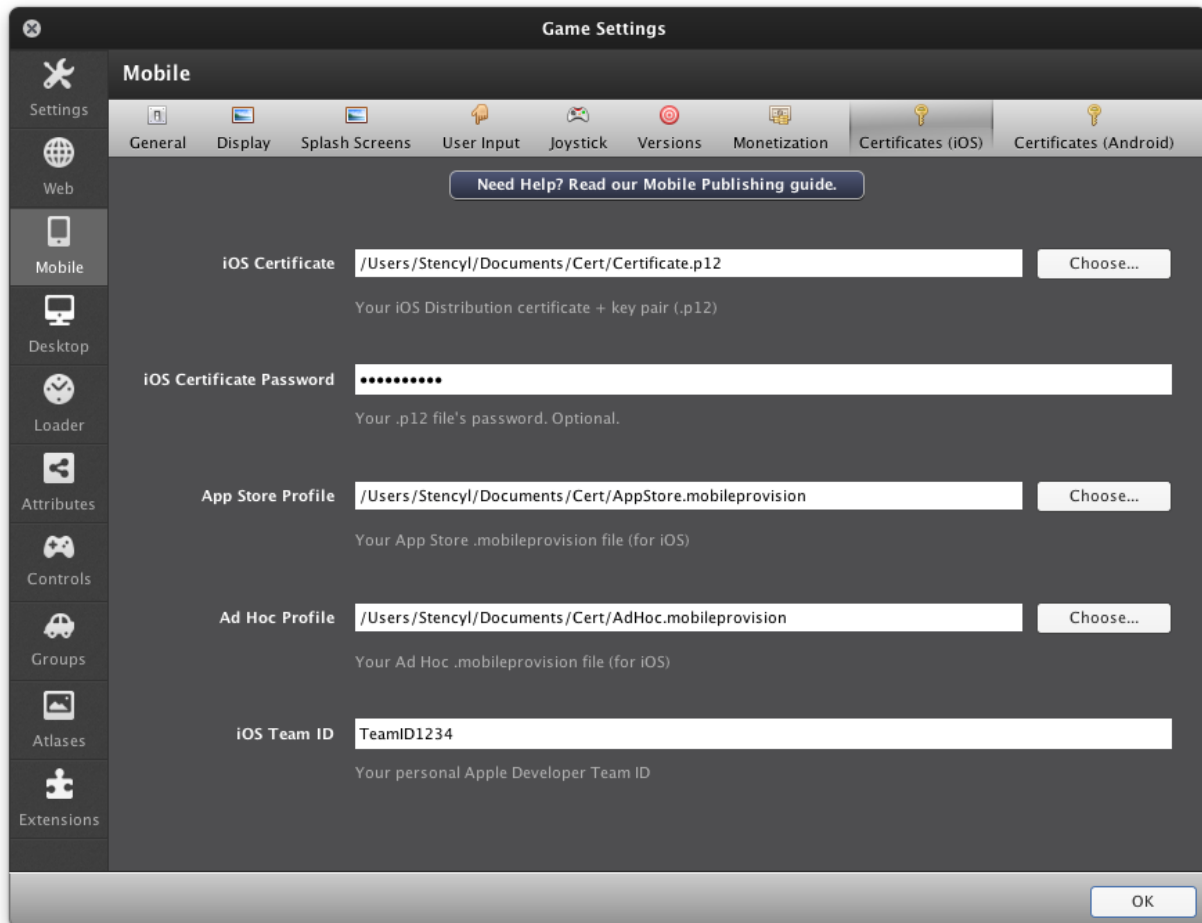


Figura 10.8: Tela com os certificados incluídos

Lembre-se de que todos os dados anteriores são gerados por meio da instalação dos arquivos gerados na plataforma Apple. Com essa configuração concluída você pode testar seu jogo em dispositivos móveis e distribuí-lo na *App Store*.

Google Play

Para distribuirmos um jogo na loja virtual da Google, primeiramente devemos acessar o site de desenvolvedores da empresa em <https://developer.android.com> e criar um conta de usuário. Após a criação da conta, será necessária a criação de uma chave de acesso, ou seja, um certificado de autorização para testes e distribuição na loja:

1. Com o jogo aberto, clique na opção **Settings** na barra superior do **Stencyl** ;
2. Na opção **Mobile** à esquerda, clique em **Certificates (Android)** :

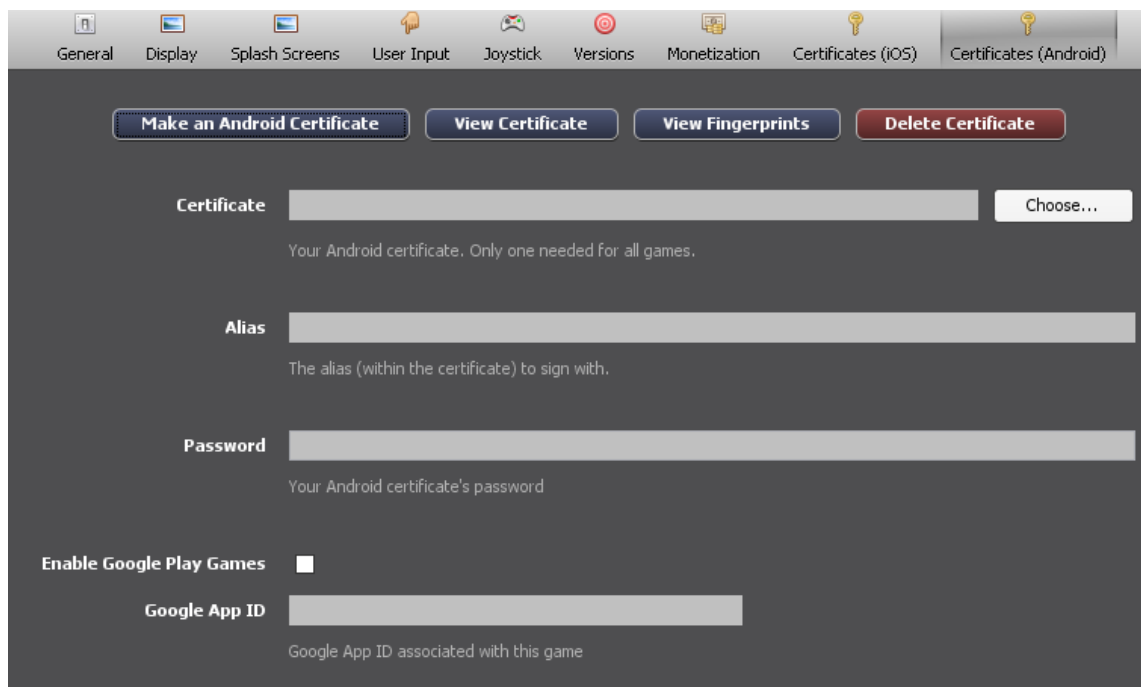


Figura 10.9: Tela para inclusão dos certificados Android

3. Para iniciar a geração do certificado, vamos clicar no primeiro botão **Make an Android Certificate** ;

Make an Android Certificate

Alias
The name for this 'certificate'

Password
6 characters minimum

First and Last Name

Company
Make something up, even if there's no 'company' for your case.

2 Letter Country Code

OK Cancel

Figura 10.10: Tela para configuração da assinatura Android

- *Alias*: aqui você digitará o nome da sua assinatura. A vantagem da plataforma Android é que basta apenas um certificado para todas as publicações na Google Play, diferentemente da plataforma da Apple que necessita de uma assinatura para cada publicação;
 - *Password*: campo para digitação da senha;
 - *First and Last Name*: primeiro e último nome do usuário;
 - *Company*: preencha este campo mesmo se não faça parte de alguma organização;
 - *2 Letter Country Code*: as iniciais do país para publicação.
4. Após o preenchimento dos campos, clique em **OK**.

Pronto! O certificado de autorização foi gerado.

DICA:

Para gerar os arquivos e subi-los às plataformas online correspondentes, vá ao menu `Publish`, desça até `Mobile` e selecione a loja desejada. As respectivas plataformas orientarão para as etapas posteriores.

10.5 Conclusão

Como vimos, o Stencyl é um programaço para aqueles que querem se iniciar no mundo do desenvolvimento dos jogos sem complicações. A compreensão e utilização dos bloquinhos de lógica em vez das linhas de código de programação facilita e muito aos leigos que se interessam pelo assunto, sejam curiosos, interessados ou até por professores e alunos em propostas mais educativas.

Como educador e estudioso do mundo dos jogos, gosto sempre de lembrar que desenvolver jogos de computador é a última etapa quando se pretende trabalhar com jogos. As fases anteriores são fundamentais para a posterior utilização da ferramenta Stencyl: planejamento, testes, adequação da narrativa e criação de personagens e cenários. Para contribuir com uma formação mais completa, seguem algumas dicas de livros fundamentais para a área. Depois, é só continuar na jornada dos jogos praticando e estudando, sempre.

- *The Art Of Game Design*, Jesse Schell (2014): bíblia da área para aqueles que desejam aprender os fundamentos do planejamento de jogos.
- *Character Development and Storytelling for Games*, Lee Sheldon (2013): livraço sobre o desenvolvimento de histórias e

personagens para o mundo dos jogos.

- *Learning Stencyl 3.X Game Development: Beginner's Guide*, Innes Borkwood (2013): principal literatura sobre Stencyl no mundo.
- *Regras do Jogo: Fundamentos do Design dos Jogos*, Katie Salen e Eric Zimmerman (2012): quatro volumes de conceitos teóricos sobre planejamento de jogos.
- *Homo Ludens*, Johan Huizinga (2014): mais teoria sobre jogos