

Maurício B. G. – mbg3dmind@gmail.com

GameMaker Studio

Exercício 3

Objetivo:

Um shooter clássico, vertical.

Recomendação: trocar cor do GM Studio para branco, deixando ícones coloridos, mais fáceis de identificar. Para isso vá no menu File – Preferences – na parte de baixo na direita da janela onde diz: Select Skin, marque GM8, então OK; por fim reinicie o GameMaker.

Variáveis:

O GameMaker possui várias variáveis pré-definidas que podemos utilizar. Existem variáveis dos objetos (cada cópia de um objeto possui estas variáveis), além de variáveis globais que não tem ligação com objetos. Podemos também criar qualquer variável que quisermos para algum fim específico.

Exemplos de variáveis de objetos:

x	(coordenada x de um objeto)
y	(coordenada y de um objeto)
hspeed	(velocidade horizontal [em pixels por step])
vspeed	(velocidade vertical [em pixels por step])
direction	(direção atual de movimento em graus [0-360; 0 é horizontalmente para a direita])
speed	(a velocidade atual nesta direção)
visible	(se o objeto é visível [1] ou invisível [0])
solid	(se o objeto é sólido [1] ou não sólido [0])

Exemplos de variáveis globais:

score	(pontuação atual)
lives	(número de vidas atuais)
mouse_x	(posição x do mouse)
mouse_y	(posição y do mouse)
room_width	(largura da sala em pixels)
room_height	(altura da sala em pixels)

Há muitas outras variáveis, tanto para objetos como globais. (estão na documentação oficial do GameMaker).

Algumas ações podem manipular o valor de certas variáveis, mas também podemos manipular diretamente.

No exemplo do nosso shooter, podemos criar uma variável que irá controlar a quantidade de disparos que a nave pode fazer. (ex: atirar a cada 5 steps do jogo)

Então a nave precisa de uma variável que vai indicar se ela pode atirar ou não.

Podemos criar a variável **pode_atirar** e no evento de criação da nave colocamos o valor 1 nesta variável (indicando verdadeiro). Assim quando o jogador quiser atirar, faremos o teste do valor da variável para ver se é permitido.

Se for permitido, criamos o tiro e colocamos **pode_atirar = 0** (impedindo que atire).

Então usamos um alarme para voltar ao valor 1 depois de 5 setps.

* Obs: este procedimento será detalhado depois.

Começando com o jogo:


Ilusão de movimento do cenário:

Uma das formas de fazer o cenário parecer que “caminha” é fazendo com que a imagem de fundo da sala (background) se movimente.

- Crie um background (water.gif)
- Crie uma sala usando este background, ela deve ter o tamanho 640x480 (aba settings, em Width e Height respectivamente). Na janela de edição da sala (aba background) coloque em Vert Speed: 2

Teste o jogo, a imagem de fundo deve estar andando.

Vamos adicionar umas ilhas no cenário para tornar mais interessante. Estas ilhas devem surgir em posições aleatórias para ficar mais natural a rolagem do cenário (não demonstrar repetição).

- Crie três sprites de ilhas (island1.gif, island2.gif, island3.gif) [removendo fundo].
- Crie três objetos usando estes sprites. No evento de criação (Create) colocaremos eles andando na mesma velocidade do background da sala. (Speed Vertical  [aba move] com speed 2). Também pode ser usado o Move Fixed, mas Speed Vertical é mais simples.
- Para ter certeza que todos os outros objetos de jogo sejam desenhados acima das ilhas, na propriedade Depth de cada ilha (logo abaixo da opção Persistent e Uses Physics) usaremos o valor 10000. (quanto mais alto o valor, mais cedo o GameMaker desenha o objeto, assim os outros aparecerão por cima)

Coloque as ilhas (1 cópia de cada) em algum local da sala e teste o jogo.

Agora iremos testar quando as ilhas saírem da tela, e então programar para que elas voltem a aparecer no topo (em uma posição horizontal aleatória).


Para isso iremos usar algumas variáveis (já pré-definidas no GameMaker):

y = posição vertical do objeto (sendo 0 = topo)


room_height = altura da sala

Então iremos testar se o **y** do objeto é maior que **room_height** (ou seja, saiu da sala).

Se for, a ilha deve ir para o topo da sala.

- Para isso no evento Step de cada ilha use a ação Test Variable  (aba control) sendo:
variable : y
value: room_height (<= copie o nome exato da variável)
operation: greater than

Agora colocaremos a ação que será executada se este teste for verdadeiro:



- Jump to Position  (aba move), para x usaremos uma função que sorteia um valor entre 0 e a largura máxima da sala (que é a variável room_width).
 - Então copie esta função: random(room_width)
- Para o atributo y desta ação podemos usar -65, para fazer a ilha aparecer acima da tela. (pois é o tamanho da altura do sprite da ilha)

Faça isso para cada ilha e teste o jogo. Se tudo deu certo, toda vez que uma ilha sai da tela, ela volta a aparecer no topo, em uma posição aleatória horizontalmente.


Avião principal:


- Crie sprite com o avião do jogador (myplane.gif) [removendo fundo]. Este sprite é composto por 3 sub-imagens que serão animadas automaticamente (hélices girando).
 - Nesta mesma janela de criação de sprite, coloque a origem no centro, clicando no botão “center”. (alguns testes usam este ponto como referência, por isso é importante estar no centro do avião; ex: tiros devem sair do centro)
- Crie objeto “jogador”, usando este sprite. Coloque Depth -100 (logo abaixo da opção Persistent e Uses Physics) para que ele seja desenhado acima de tudo.


Para movimentação do avião, devemos tomar cuidado para ele não sair da tela. Então vamos testar se a variável “x” é maior que um valor próximo da borda (ex: 40) e permitir movimentação somente se isto for verdadeiro. (neste caso para esquerda)

- No evento Keyboard – Left, coloque ação Test Variable  (aba control) sendo variável: x ; valor: 40 e operação “maior que” (greater than).
- Coloque outra ação logo abaixo: Jump to Position  (aba move), x: -4 ; y: 0 ; relative ligado. Isto fará com que o avião pule ligeiramente para o lado esquerdo, enquanto a tecla direcional esquerda estiver sendo pressionada.


Faça o equivalente para as outras direções:

Keyboard – Right, testando se x é menor (less than) que a largura da sala menos uma pequena margem para o avião não bater na borda “**room_width – 40**” (<= copie exatamente). Depois Jump to Position  com x: 4 (relative ligado).


Keyboard – Up, testando se **y** é maior que 40. Depois Jump to Position  com y: -2 (relative ligado). Note que diminuindo o valor de y o avião sobe. Usamos valor 2 na vertical para coincidir com a velocidade de rolagem do background.

Keyboard – Down, testando se **y** é menor que a altura da sala menos uma pequena margem para o avião não bater na borda “**room_height – 120**” (<= copie exatamente). Usamos um valor maior aqui para o avião ficar mais longe da base da tela, pois adicionaremos uma barra de interface depois. Insira outra ação Jump to Position  com y: 2 (relative ligado).

Tiro:


- Crie sprite (bullet.gif) [removendo fundo], coloque a origem no centro.
- Crie um objeto usando este sprite. No evento de criação faça o objeto andar para cima com o Vertical Speed , sendo speed: -8 (aba move).

Agora devemos destruir o tiro quando ele sair da tela, para poupar processamento. Há um evento extra que indica quando um objeto saiu completamente da sala:

- Crie o evento Other – Outside room, coloque a ação Destroy Instance  (aba main1).




Nota: Sempre que um objeto sair da sala e não for mais necessário, ele deve ser destruído para liberar memória, caso contrário o jogo irá ficar cada vez mais lento.

Para o avião atirar com a barra de espaço:

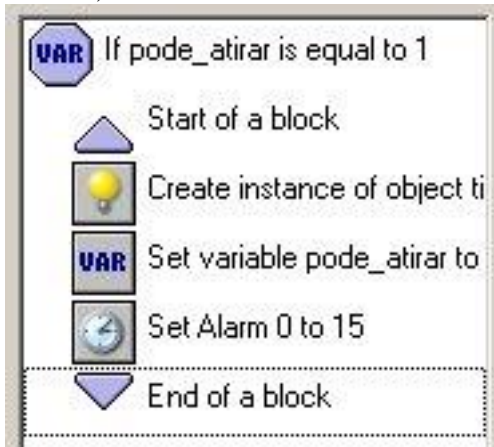
- Dentro do objeto jogador (que é o avião), crie o evento Keyboard – Space, coloque a ação Create Instance  (aba main1). Configure a ação para criar o objeto tiro, na posição x: 0 e y: -16 com relative ligado. Isto irá criar uma cópia do objeto tiro, sobre o centro do avião (posição relativa).

Teste o jogo e veja que estão saindo muitos tiros quando pressiona a barra de espaço.

Para corrigir isso, iremos usar uma variável “**pode_atirar**”. Se ela for =1 podemos criar um tiro, caso contrário não. Assim que o primeiro tiro é criado, **pode_atirar** ficará com o valor 0, não permitindo que saia mais nenhum tiro. Então criamos um alarme que em meio segundo irá trocar o valor de **pode_atirar** para 1 novamente.

- Para implementar isso, o primeiro passo é só atirar se **pode_atirar** for = 1. Então no evento já existente <Space> do avião, **antes** de criar um tiro colocamos a ação Test Variable  (aba control). Configuramos para testar se **pode_atirar** é igual a 1.
- Depois da ação do tiro (que já existe) iremos trocar o valor de **pode_atirar**. Utilize a ação Set Variable  (aba control) para deixar **pode_atirar** = 0
- Agora iremos criar o alarme. Use Set Alarm  (aba main2), com steps: 15 (meio segundo)

Importante: Como estas ações só devem ser executadas se a condição do teste for verdadeira, devemos colocar o início e final de bloco (Start Block e End Block – aba control) entre elas. Desta forma:



- Então quando o alarme disparar, devemos trocar o valor de **pode_atirar** para 1 novamente. (adicione **evento** Alarm 0, após adicione ação Set Variable)
- Não podemos esquecer que no início do jogo, quando o avião é criado, **pode_atirar** deve ser = 1, para poder dar algum tiro e não dar erro ao testar a variável. Então no evento de criação use Set Variable para fazer isso.

Teste o jogo.

Inimigo:

O primeiro inimigo será simples, só voará para baixo da tela. Se colidir com o avião do jogador, o jogo acaba.



- Crie sprite (enemy1.gif) [removendo fundo e coloque a origem no centro] e um objeto que o use. No evento de criação coloque Speed Vertical (aba move) com valor 4. Temos que fazer o avião voltar a aparecer na tela quando ele sair por baixo (semelhante ao comportamento das ilhas).
- Um meio alternativo de fazer isso é usando o evento Other – Outside Room, que acontece assim que o objeto sair completamente da sala. Coloque ação Jump to Position (aba move); no atributo x: random(room_width) ; no atributo y: -16

Adicione o inimigo na sala e teste.




Agora precisamos fazer ele colidir com o tiro e morrer; além de colidir com o avião do jogador e acabar o jogo. Primeiro com o tiro:

- Crie o sprite (explosion1.gif) [removendo fundo e coloque a origem no centro] e um objeto para ele. (pode chamar de explosao_inimigo) Essa animação de explosão será


tocada quando o inimigo for atingido. Também podemos criar um som de explosão (snd_explosion1.wav)

- No avião inimigo, adicione o evento colisão com o tiro. Então use as seguintes ações:
 - Tocar o som de explosão, Play Sound  (aba main1).
 - Fazer desaparecer o tiro: Destroy Instance  (aba main1) – opção **Other** (isto fará destruir a **outra cópia** do objeto envolvido na colisão, no caso o tiro; impedindo que ele siga destruindo outros inimigos que encontrar)






- Mostrar animação de explosão: Create Instance  (aba main1), coloque o objeto que é a explosão e ligue relative. Deixe x,y em 0, pois queremos a explosão exatamente sobre a posição relativa do avião.
- O avião inimigo não deve morrer, ele deve surgir novamente no topo da tela, em alguma posição em x; então use: Jump to Position  (aba move); no atributo x: random(room_width) ; no atributo y: -16
- Agora podemos aumentar a pontuação com Set Score  (aba score); valor 5 e relative ligado.

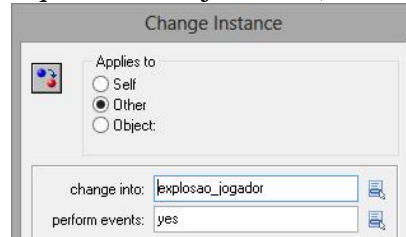
Teste o jogo. O avião inimigo deve explodir quando atingido e surgir novamente no topo da tela. Porém a explosão **não está desaparecendo!**



- No objeto explosão, use o evento Other – Animation End (é executado quando a animação chegar no fim); ali coloque um Destroy Instance  (aba main1), para a explosão “morrer”. (deixe a configuração padrão: self, para o próprio objeto morrer)

Agora a colisão do avião inimigo com o avião do jogador:

- Crie um som de explosão diferente (snd_explosion2.wav) e o sprite para explosão (explosion2.gif) [removendo fundo e coloque a origem no centro]. Esta explosão é maior que a outra. Crie um objeto que a use. (pode chamar de explosao_jogador para diferenciar)
- Neste objeto explosão, no evento Other – Animation End, iremos fazer o jogo acabar:
 - Primeiro a explosão deve sumir no final da animação: Destroy Instance 
 - Por fim reiniciar o jogo: Restart Game  (aba main2)
- No objeto do avião inimigo, crie um evento de colisão com o jogador e use:
 - Change Instance  (aba main1), isto irá trocar um objeto por outro, iremos usar para o avião do jogador ser trocado pela explosão (configure para o objeto explosao_jogador em “change into:”). **Importante: Marque Other**





ao invés de Self (caso contrário quem irá virar explosão é o avião inimigo, e não o jogador como queremos; veja abaixo)



- Play Sound  (aba main1) para tocar o som de explosão
- E o avião inimigo também deve morrer: Destroy Instance  - Self

Teste o jogo.

Ele está jogável, porém só tem um avião inimigo por vez na tela. O jogo pode ficar mais interessante com mais inimigos ao mesmo tempo. Uma forma eficiente de fazer isso é criando um objeto especial, não visível no jogo, que irá controlar a criação de inimigos.


- Crie um objeto (controlador_inimigo) e desligue a opção “visible”. Ele não precisa de sprite.
 - No seu evento de criação, crie um inimigo no topo da tela com posição x aleatória: Create Instance  (aba main1); x: random(room_width) ; y: -16
 - Depois coloque um alarme: Set Alarm  (aba main2), com 200 steps
- No evento alarm0, que será usado quando o alarme disparar, coloque outro inimigo e configure o alarme novamente.
 - Create Instance  igual a anterior, com x: random(room_width) ; y: -16
 - Set Alarm , com 500 steps

Coloque este objeto na sala e teste o jogo. Agora vários inimigos podem aparecer ao mesmo tempo, pois quando o alarme dispara, cria mais um inimigo, e se configura para disparar de novo, fazendo um ciclo infinito.

Vidas, pontuação e dano:

Para melhorar o jogo, podemos fazer a colisão com o inimigo gerar um dano no avião do jogador. Assim teremos uma barra de energia na tela, sendo que o avião só será destruído se o dano for o máximo possível. Além disso, podemos fazer ele ter três vidas. Tudo isso será mostrado em uma barra de status (interface) na base da tela.





- Crie um objeto (controlador_vida). Usaremos o evento Draw, que determina o que será desenhado na tela. (obs: as ações que estão na aba draw só funcionam dentro deste evento)
 - Neste evento iremos desenhar na base da tela a imagem de interface, para isso crie um sprite (bottom.gif), que não será transparente. De volta ao

evento Draw do controlador_vida, use Draw Sprite  (aba draw) indicando o sprite da interface, e as coordenadas x: 0; y: 404.




- Para ter certeza que será desenhado acima de todos os outros objetos, coloque -10000 em Depth no controlador_vida. (logo abaixo da opção Persistent e Uses Physics)

Adicione este objeto na sala e rode o jogo.







Nesta interface será mostrada a pontuação, numero de vidas e a barra de energia. Então de volta ao evento Draw deste controlador_vida, use:

- Set Color  (aba draw), para determinar a cor que será usada, escolha um tom de amarelo que combine com a interface.
- Draw Score  (aba score), com as coordenadas x:180; y: 440 (sem caption, pois já está escrito "score" na imagem da interface)
- Draw Health  (aba score), desenha uma barra de vida ou energia, x1: 12; y1: 449; x2: 138; y2: 459. **Obs: não confunda com a ação Score Caption (mesmo ícone)**
- Por último iremos desenhar as vidas como imagens (pequenos aviões). Para isso crie o sprite (life.gif) [sem fundo]. De volta ao evento Draw do controlador_vida, use a ação Draw Life Images  (aba score) com as coordenadas x: 16; y: 410

O que falta para nossa interface funcionar é determinar no início do jogo quantas vidas o jogador tem, a quantidade máxima de energia, e deixar a pontuação zerada.




- No evento Create do controlador_vida, use:
 - Set Score  (aba score), valor 0 (sem relative).
 - Set Lives  (aba score), valor 3 (sem relative).
 - Set Health  (aba score), valor 100 (sem relative).

Agora a interface funciona, **porém o jogo ainda está acabando quando o jogador bate em alguém**. Precisamos mudar isso, quando bater o jogador deve só perder energia. E então irá morrer só se a energia chegar a zero.



- Deve ser alterado o evento de **colisão do avião inimigo com o jogador**. Agora ele não deve matar o jogador, só tirar energia.
 - Deixe a ação que toca o som de explosão .
 - **Apague** a ação que troca o objeto do jogador para a animação de explosão (Change Instance )
 - **Apague** a ação que destrói o avião inimigo (Destroy Instance )
- Quando o inimigo bater no avião do jogador, ele irá criar a explosão menor, ir de volta ao topo da tela e diminuir a energia do jogador. Então use:
 - Create Instance  (aba main1), configure para a explosao_inimigo e posição **relativa** x:0, y:0
 - Jump to Position  (aba move), x: random(room_width); y: -16
 - Set Health  (aba score), valor -30, relative ligado.

Ok, tudo funciona, porém o avião do jogador não está morrendo nunca...



Para fazer isso, usaremos eventos especiais que são acionados quando a energia chegar a zero ou não houver mais vidas restantes.

- Dentro do controlador_vida, use o evento Other – No More Health. Quando a energia acabar, vamos tocar o som de explosão, trocar o objeto do jogador para a explosão e deixar a energia de novo em 100%; para isso use:
 - Play Sound  (use o som de explosão mais forte (2))
 - Change Instance  (aba main1), use **Applies to: Object** e aponte para o avião do jogador. (em “change into:” explosao_jogador)
 - Set Health  (aba score), valor 100. (sem relative)

Agora o avião está morrendo quando acaba energia, mas o jogo já acaba. Ele deve só perder uma vida, e continuar:

- Dento do objeto explosao_jogador, evento Animation End:
 - Deixe a ação Destroy Instance, para a explosão desaparecer.
 - **Retire** a ação Restart Game
 - Vamos criar uma nova cópia do objeto avião do jogador: Create Instance  (aba main1), posição x:0, y:0, relative ligado. (ou seja, será no mesmo lugar onde o avião estava antes)
 - Para diminuir uma vida: Set Lives  (aba score), valor: -1, relative ligado.

Ok, agora o avião morre quando a energia acaba, perde uma vida e continua jogando. **Porém mesmo perdendo todas as vidas, o jogo não acaba nunca!** Para resolver isso vamos usar o evento especial - No More Lives:

- No objeto controlador_vida, crie o evento Other – No More Lives, use:
 - Para mostrar uma mensagem de “Game Over” quando acabar o jogo, use a ação Display Message  (aba main2)
 - Restart Game  (aba main2)

Agora o jogo está totalmente jogável e funcional.






Para deixar mais interessante, podemos criar outros inimigos, com comportamento diferente (ex: eles atiram).

Mais inimigos:

Como o novo inimigo é parecido com o já existente, duplique o objeto inimigo1 (clique com botão direito sobre objeto e Duplicate) e altere a cópia colocando um novo nome e novo sprite. Para isso crie um sprite (enemy2.gif) [sem fundo e coloque origem no centro].



Este inimigo é mais perigoso, então faremos ele dar 10 pontos ao jogador quando acertado. Na colisão dele com o tiro, altere a ação da pontuação para dar 10 pontos ao invés de 5.

Para o inimigo atirar, precisamos de um novo sprite (enemy_bullet1.gif) [sem fundo e coloque a origem no centro] e um objeto que o use, chame de tiro_inimigo. Então adicione:

- No evento de criação, use Speed Vertical  (aba move) para o tiro já nascer em movimento (speed 8).
- Ele deve “morrer” quando sair da sala. Então no evento Other – Outside Room, use Destroy Instance  (aba main1).
- Quando o tiro colidir com o avião do jogador, deve tocar som de explosão, desaparecer, diminuir a energia do avião em -5. Então no evento colisão com o jogador use:
 - Play Sound  (aba main1);
 - Destroy Instance  (aba main1);
 - Set Health  (aba score), valor -5, relative ligado.


A programação do tiro do inimigo está pronta. Agora precisamos fazer o inimigo2 atirar. Podemos fazer a decisão de atirar ou não do inimigo ser algo aleatório. Uma forma de fazer isso é usar a ação **Test Chance** (aba control). Esta ação cria um “dado virtual” com o número de lados especificado. Então este dado é “jogado” e se cair o número 1 a próxima ação é executada. Ou seja, quanto mais lados o dado tiver, menor a chance de cair 1. Por exemplo, se quisermos que a ação tenha 50% de chance de ser executada, criamos um dado com 2 lados. Se quisermos uma chance menor, aumentamos o número de lados.

Podemos usar este teste no evento **step**, que fará o teste a cada “passo” do jogo (lembrando que 30 passos = 1 segundo). Assim um dado com 30 lados terá em média uma condição verdadeira a cada 30 passos. (ou seja, 1 tiro a cada segundo)

- No evento step do avião inimigo2, crie a ação Test Chance  (aba control), com 30 lados. Logo após coloque a ação Create Instance  (aba main1), configurando para criar um tiro inimigo na posição x:0 ; y:16 com relative ligado.

Para colocar este inimigo no jogo, podemos utilizar o objeto controlador_inimigo:

- No evento de criação crie um novo alarme (Set Alarm 1  [aba main2]) com 1000 steps.

- Adicione o **evento** Alarm1, crie uma cópia do inimigo2 em uma posição x aleatória, de forma semelhante ao que foi usado no Alarme 0.
- Então coloque novamente a ação Alarme 1  com 500 steps.

Desta forma o inimigo 2 deve aparecer depois de aproximadamente 30 segundos de jogo, e voltar a aparecer a cada 15 segundos.

Atirando na direção do jogador

Iremos agora criar outro tipo de inimigo, que dispara um tiro que vai na direção do jogador. Para isso duplique o objeto inimigo2 e crie um novo sprite (enemy3.gif) [sem fundo e coloque a origem no centro]. Use então este sprite para o inimigo3.

Este inimigo é mais perigoso que os outros, então daremos ao jogador 20 pontos quando destruí-lo. Ajuste a pontuação na ação respectiva, no evento de colisão com o tiro.


O tiro do inimigo3 será especial, então vamos usar um sprite diferente (enemy_bullet2.gif). Duplique o objeto tiro_inimigo e use este sprite. Troque o nome para tiro_inimigo2.

Vamos configurar o objeto inimigo3 para atirar este tiro novo:

- No evento step, na ação que rola um “dado” virtual, troque o número de lados para 80, pois este tiro é muito difícil de desviar, se tiver vários ao mesmo tempo na tela será muito complicado para o jogador.
- Na ação que está logo abaixo, configure para criar o tiro_inimigo2.

Então agora o avião inimigo 3 lançará em média 1 tiro a cada 80 passos (2,6 segundos).

Vamos programar o comportamento do tiro_inimigo2:





- No evento de criação, use: Move Towards  (aba move), esta ação movimenta em direção a um ponto. Este ponto em questão será a posição do avião do jogador. Para isso podemos usar as variáveis **jogador.x** e **jogador.y**, com speed 8.

Obs: Para acessar o valor de variáveis de outro objeto, podemos usar *nome_objeto.variável*. Neste caso, como queremos variáveis de posição x;y usamos *nome_objeto.x* ; *nome_objeto.y*

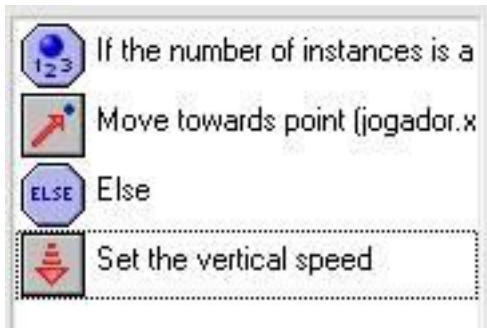
Quando há várias cópias do mesmo objeto na tela, é retornada a informação referente a primeira cópia criada. Quando não há nenhuma cópia, é gerado um erro.


Isto pode ser um problema, pois no momento que o avião do jogador é destruído não há nenhuma cópia deste objeto no jogo, e se o inimigo3 tentar atirar neste mesmo instante, teremos um erro.


Podemos resolver isto testando se existe o avião do jogador na tela, e só então permitindo que o tiro seja lançado. Há uma ação que pode contar o número de cópias (instances) de um determinado objeto. Então caso não existir nenhuma cópia, para evitarmos o erro, o tiro seguirá em vertical para baixo, sem seguir nenhum ponto.

- Retomando: no evento de criação do tiro_inimigo2, teremos:
 - Test Instance Count  (aba control). Configure objeto: jogador, number: 0, operation: Larger than. Ou seja, se o objeto jogador tiver mais que 0 cópias, executará a próxima ação.
 - **Já criado:** Move Towards : x: jogador.x ; y: jogador.y ; speed: 8
 - Para especificar a ação a ser feita se o teste for negativo, use **Else**  (aba control).
 - Por fim, aqui irá a ação se o teste falhar: Speed Vertical  (aba move), com speed: 8 (esta ação já existia quando foi feita cópia do tiro_inimigo)

Ex:



Agora só precisamos adicionar este novo inimigo ao jogo. Podemos usar novamente o controlador_inimigo, adicionando no evento de criação mais um alarme . (Alarm2 = 2000 steps).

Então adicione o **evento** Alarm2, copie as ações que estão nos outros alarmes e modifique para criar o inimigo3 e reajustar o alarme 2  em 1000 steps.

(Desta forma, o primeiro inimigo3 deve ser criado em 66 segundos de jogo, e os próximos a cada 33 segundos).



Usando time lines (linha do tempo)

O jogo na versão atual está funcionando e dá algum desafio ao jogador, pois fica mais difícil com o tempo. Porém não temos controle sobre a criação dos aviões inimigos, eles podem surgir em qualquer posição, até uns por cima dos outros, sem nenhuma formação interessante. Além disso, eles não desaparecem nunca, sempre retornam a todo momento.




Seria melhor para o jogo ter um controle maior sobre a criação dos inimigos, possibilitando a construção de desafios mais elaborados. Quando a criação dos inimigos são em locais definidos, podemos criar formações dos aviões (ex: voando em V) e permitir que o jogador fique mais experiente no jogo, pois cada vez que joga os aviões aparecerão no mesmo lugar. Isso permite que ele treine a cada jogada e melhore sua performance.

Para criar eventos específicos que ocorrem com o passar do tempo podemos usar um outro recurso do GameMaker, chamado “time lines”. Nesta linha do tempo podemos criar ações a cada momento (steps) que quisermos.

Teremos que ajustar o jogo para usar a time line. Como ela que irá controlar a criação dos inimigos, vamos modificar o jogo para que cada inimigo destruído realmente “morra” e não volte a aparecer no topo da tela em outra posição.

- Para isso, no objeto de cada inimigo, no evento colisão com o jogador, retire a ação Jump to Position  e coloque Destroy Instance  no lugar (aba main1).
 - No evento colisão com o tiro, faça a mesma coisa.
 - No evento Outside Room, faça a mesma coisa.

Vamos então criar uma time line e configurar para ela criar os inimigos:

- Crie uma nova time line (no menu geral na esquerda). Coloque o nome que quiser.
- Adicione um momento de tempo (Add – 0). Use Create Instance  (inimigo1; x:320 y: -16). [ou seja, no step 0 cria inimigo1 no meio da tela em x, e acima da tela em y)
- Adicione outro momento de tempo (Add – 100). Agora vamos criar 2 inimigos, um no lado do outro. Então use Create Instance  (inimigo1; x:200 y: -16), e mais outro Create Instance  (inimigo1; x:400 y: -16).

Porém uma time line sozinha não faz nada, devemos vincular uma time line a um objeto, que então fará ela executar.

Podemos usar o controlador_inimigo para isso. No seu evento de criação use:

- Set Time Line  (aba main2). (**retire todas as outras ações e eventos**)


Nota: A time line pode ser configurada para ficar cíclica, infinita (sempre que acaba, começa de novo). Basta colocar Loop, na configuração.

Teste o jogo. Deve surgir um inimigo no centro da tela, e depois de 100 steps mais dois inimigos, um no lado do outro.

Crie novos momentos de tempo na time line, gerando novos inimigos. Planeje o fluxo e as formações de entrada de inimigos na tela, criando uma experiência interessante para o jogador.

(Atenção: se o avião inimigo for criado completamente fora da sala de jogo ex: y: -32, o evento Outside Room entrará em ação. Se lá estiver a ação Destroy Instance, o avião será destruído antes de entrar na sala.

Nesse caso não devemos usar o evento Outside Room para destruir o avião quando sair da sala, e sim usar o evento Step, testando se sua posição em y está maior que a altura da sala.

Desta forma: Test Variable  (variable: y ; value: room_height ; operation: greater than)

Então se isto for verdadeiro, o avião é destruído. Se for falso, ele joga o “dado” para ver se atira; pois este seria seu comportamento normal no evento step, no caso do inimigo 2 e 3)