Arcade Game Designer

Copyright Jonathan Cauldwell, 2008-2017

Instruções para a versão 4.7

Como o PGD e o SEUD, o Arcade Game Designer é uma ferramenta para escrever os seus próprios jogos de arcada simples. Embora seja um pouco mais complexo do que esses 2 utilitários e não o ajudará a criar shoot-em-ups de rolagem, ele deve ser capaz de produzir uma ampla variedade de assuntos de arcade com o mínimo de confusão, contanto que você seja paciente e experiente, preparado para ficar nele. Além disso, os jogos produzidos usando o utilitário são independentes e podem ser distribuídos gratuitamente.

A AGD já foi bem utilizada por vários autores e é responsável por muitos jogos. Alguns exemplos são Apulija-13, Hedgehogs, Kyd Cadet, Land of Mire Mare e Trooper: ponto 5. Todos estes são completamente recomendados e podem ser encontrados nos arquivos do World of Spectrum se você realmente quiser ver o que versões anteriores do AGD podem fazer.

Isso não é tudo, no entanto, AGD agora vem com explosões, trilhas mágicas de poeira/vapor, sprites maiores, lasers, rolagem de mensagens de texto e uma facilidade para colorir sprites, dando ao designer de jogos a habilidade de criar jogos que são mais profissionais do que nunca. É facilmente capaz de produzir jogos para o covertape ou padrões de software de orçamento de tempos passados.

Agora às más notícias. O AGD tem sua própria linguagem integrada, um caso bastante simplista com funções simples e algumas variáveis embutidas que você terá que pegar se desejar adicionar seus pequenos detalhes, lógica do jogo ou desviar-se do construtor, em modelos. Na verdade, o léxico não é grande, é principal mente uma questão de entender como usar cada comando ou função em particular. É principalmente muito direto. Enquanto o programa informará se ele encontrar uma palavra que não entende, a verificação de sintaxe é inexistente. No entanto, o AGD 4 moverá o cursor diretamente para a linha em questão, para que você, pelo menos, saiba onde está o problema.

Porque eu queria produzir um designer de jogos para todos usarem, o programa é gratuito para baixar do meu site. Os direitos autorais, no entanto, permanecem com o autor. Doações não são necessárias, por isso, se você realmente precisa mostrar seu apreço, compre uma cópia do SEUD ou PGD em www.cronosoft.co.uk ou algo assim, se isso falhar, os planos estão em andamento para lançar uma versão AGD da fita no rótulo Cronosoft em um futuro próximo.

Embora AGD 4 é a versão mais robusta lançada até o momento, o programa vem como você o encontra, sem garantias de que ele é totalmente livre de bugs. Por favor, não espere suporte por e-mail - terei prazer em responder a perguntas e ouvir solicitações de melhorias e correções de bugs nos fóruns do AGD. De tempos em tempos, recebo atualizações, dependendo de outros projetos e da vida em geral. Até lá, salve seu trabalho sempre que possível e, de preferência, como um instantâneo, se estiver usando um emulador.

Mais importante de tudo, divirta-se. Enquanto isso, vou a Ripley para me embebedar.

Jonathan Cauldwell, 4th April 2008.

©zxspectrumdev

http://www.spanglefish.com/egghead/

Construindo um Game

AGD requer um pouco de troca de fitas para salvar e carregar, por isso, se você estiver usando um emulador, é uma boa ideia desligar a opção de carregar automaticamente os arquivos de fita assim que forem inseridos. Não fazer isso pode resultar na perda de qualquer trabalho que você tenha feito em um jogo caso tente alterar a fita.

Em SPIN, clique em Ferramentas e selecione Opções no menu. Selecione Arquivos na lista à esquerda e, em seguida, clique na guia "**Fitas**" na parte superior. Certifique-se de que não haja uma marca na caixa ao lado de Carregar fitas automaticamente. Agora você está pronto para começar a desenvolver com o AGD.

Basicamente, os jogos são uma combinação de diferentes elementos:

* blocos de caracteres são combinados para formar telas
* sprites (incluindo o jogador) são posicionados em suas posições iniciais em cada tela, e atribuida uma característica de tipo
* lógica e então escrita para cada tipo de sprite - jogador, inimigos, objetos etc.

Depois disso, é uma questão de adicionar mais alguma coisa que você queira.

Main Menu

Permite selecionar uma função do jogo que você deseja projetar. A quantidade de memória restante disponível para seu aplicativo é exibida no canto superior direito da tela.

Window area

Você pode variar o tamanho e a posição da área de reprodução na janela / opção de rolagem.

As teclas do cursor permitem posicionar a janela em qualquer lugar da tela.

Alterar o tamanho da janela e/ou a direção de rolagem pode alterar a quantidade de dados necessária para cada coluna ou linha de blocos em cada tela e, portanto, invalidar os níveis atuais, se este for o caso, o programa perguntará se você deseja destruir os dados do mapa antes de prosseguir. É uma boa ideia selecionar primeiro o tamanho da área de jogo e ficar com ela.

**1** = janela estreita.

**2** = alargar janela.

**Q** = encurtar a janela.

**A** = alongar janela.

Palette

Desde a versão 3, a AGD tem suportado espectros e emuladores habilitados para ULAplus por padrão. Se você estiver usando a versão 0.7s do SPIN, poderá ativar a emulação do ULAplus selecionando Opções no menu **Ferramentas** (ou apenas pressionando **F8**) e, em seguida, selecionando a opção **Exibir** à esquerda. Se você selecionar a guia **Emulação** na parte superior, deverá haver uma caixa de seleção para 64 ULA aprimorados por cor. Certifique-se de que está selecionado e clique no botão **OK** na parte inferior.

O editor de paleta ULA+ permite que você mova as 64 cores das quatro tabelas de consulta de cores (CLUTs), modificando-as. O valor RGB da cor selecionada é mostrado no canto inferior direito da tela.

**R** = muda o componente vermelho da cor.

**G** = muda o componente verde da cor.

**B** = muda o componente verde da cor.

**ENTER** = retornar ao menu principal.

Keys

Isso permite que você defina as chaves que seu jogo usará. São permitidas 7 chaves e o joystick Kempston é mapeado nos cinco primeiros. A ação Kempston equivalente é mostrada na tela quando você é solicitado para cada chave. AGD padrão para o teclado, mas se você está tecnicamente ocupado, você pode querer usar o joystick Kempston dentro do seu jogo via POKE 32005,1 ou joystick Sinclair com POKE 32005,2. Para teclado, POKE 32005 com qualquer outro valor.

Character Block Design

Blocos são usados para construir as telas. Cada bloco tem 8x8 pixels de tamanho e tem um tipo diferente atribuído a ele. Existem vários tipos diferentes, cada um com seu próprio conjunto diferente de atributos. Esses são:

EMPTY SPACE (ESPAÇO VAZIO) - o jogador se move livremente através de blocos de espaço livre.

NORMAL PLATFORM (PLATAFORMA NORMAL) - o jogador se move livremente pelas plataformas da esquerda, direita ou abaixo.

SOLID WALL (PAREDE SÓLIDA) - um bloco que é intransponível de qualquer direção.

LADDER - o jogador se move livremente através de blocos de escada em qualquer direção.

FODDER - como parede sólida, pode ser removido com certas funções.

DEADLY - sprites passam por isso, testes de função DEADLY para contato.

CUSTOM - sprites passam por isso, testes de função CUSTOM para contato.

Mova o cursor pelo personagem com as teclas do cursor. Use **SPACE** ou **0** para definir / desmarcar um pixel. **ENTER** retorna ao menu principal. Você pode definir as cores com **P**, **I** e **B**. Como bits brilhantes e Flash podem ser definidos ou não definidos, então se você pretende que seu jogo seja executado em Spectrum modificados com ULA+ ou emuladores, todos os 4 CLUTS estarão disponíveis.

**Q** = mover para a esquerda através da lista de propriedades do bloco.

**W** = mover para a direita através da lista de propriedades do bloco.

**L** = último bloco.

**N** = próximo bloco.

**P** = cor do papel.

**I** = cor da tinta.

**B** = alterar o brilho e os bits de flash.

**M** = copiar o bloco atual para a área de transferência.

**K** = colar o bloco da área de transferência.

**C** = limpar o bloco atual.

**X** = cria um novo bloco de caracteres.

**D** = excluir o bloco de caracteres atual.

**ENTER** = retornar ao menu principal.

Screen Layout

Mova o cursor pela tela com as teclas do cursor. Use **1** e **2** para selecionar o bloco de caracteres que você deseja colocar na posição atual do cursor. **ESPAÇO** ou **0** coloca o bloco desejado na tela atual nessa posição. Alternativamente, você pode querer usar o modo de desenho rápido, pressionando **F**. Isso coloca automaticamente o bloco selecionado na posição atual do cursor toda vez que você mover o cursor. Para sair do modo de desenho rápido e retornar para a colocação manual de blocos, pressione a tecla **F** novamente. Você pode copiar uma tela. Pressione **M** para marcar a tela que você deseja copiar, então vá para a tela que você deseja copiar e pressione **K** para colar, **ENTER** retorna ao menu principal.

**1** = mover para a esquerda através da tabela de blocos.

**2** = mover para a direita através da tabela de blocos.

**F** = alternar modo de desenho rápido.

**M** = marcar a tela atual.

**K** = colar da tela marcada.

**N** = próxima tela.

**P** = tela anterior.

**X** = criar uma nova tela.

**D** = excluir tela atual.

**ENTER** = retornar ao menu principal.

Sprite Images

Sprites são imagens de 16x16 pixels que compõem partes móveis como o jogador, as balas dos jogadores, as naves inimigas, etc. Sprites podem ter qualquer número de quadros.

Movimente-se pela grade com o cursor, fixando e excluindo pixels com **SPACE** ou **0**. Como a detecção de colisão do AGD é baseada em coordenadas, é uma boa idéia preencher o máximo possível da área de 16x16, especialmente para sprites de jogadores e inimigos. Dê uma olhada no Monty Mole, Egghead ou BLOB da Starquake como exemplos de sprites que preenchem uma área de 16x16. Colecionáveis não importam tanto, os jogadores raramente notam ou se importam se pegam um colecionável de 3 pixels de distância.

O editor de sprites também possui uma tela de rascunho, você pode carregar arquivos SCREEN$ da fita usando a tecla **L**. você pode então mover um cursor de 16x16 pixels ao redor desta tela, então pressione **G** para pegar a imagem da tela naquele ponto e copiá-la para a área de transferência, isso pode ser colado em um quadro de sprite com a tecla **K**.

Esteja avisado, alguns emuladores carregam automaticamente arquivos de fita assim que você insere uma fita, o que removerá o AGD da memória junto com o jogo que você estava criando. Certifique-se de ter desativado esta opção no seu emulador primeiro.

**X** = inserir Sprite.

**D** = excluir Sprite.

**C** = grade de Sprite claro.

**G** = pegue o sprite da tela do bloco de notas e copie para a área de transferência.

**L** = carrega a tela do bloco de notas.

**M** = copiar sprite ou tile para a área de transferência.

**K** = copiar sprite ou tile da área de transferência.

**H** = inverte o sprite horizontalmente.

**V** = inverte o sprite vertical mente.

**N** = próximo sprite.

**P** = sprite anterior.

**I** = inserir quadro.

**R** = remover quadro.

**F** = próximo quadro.

Sprite Positions

Isso permite que você posicione sprites em suas posições iniciais para cada nível usando as teclas do cursor. 8 tipos de sprites estão disponíveis, e sprites podem ser configurados como qualquer um destes, permitindo que você posicione bônus, sprites inimigos, a posição inicial do sprite do jogador ou qualquer outro tipo de sprite que você tenha configurado. As imagens do sprite são separadas dos tipos de sprite. Imagens são apenas as imagens que você desenha no editor de sprites, enquanto o comportamento de um sprite é definido por seu tipo. O tipo 0 é geralmente usado para o sprite do jogador (ou sprites), embora você possa mudar isso. Você precisará alocar um tipo para cada item em movimento diferente, como alienígenas, marcadores ou itens colecionáveis, e determinar seus movimentos usando uma linguagem de script simples. Veja a seção sobre eventos para obter informações sobre sprites em movimento.

**N** = próxima tela.

**P** = tela anterior.

**Q** = move próximo sprite.

**I** = mudo a imagem do sprite.

**T** = altera o tipo de sprite.

**D** = excluir sprite atual da tela.

**X** = adiciona novo sprite a tela.

Objects

Distinto dos sprites, os objetos são itens estáticos que um sprite pode pegar ou possivelmente soltar, como os sprites, eles consistem em imagens de 16x16 pixels, embora tenham apenas um quadro. Estes podem ser editados da mesma maneira que as imagens de sprites.

Para cada objeto, você precisará definir sua posição no início do jogo. Ao contrário dos sprites, os objetos não reaparecem toda vez que o jogador entra em uma sala, então eles podem ser pegos em uma sala e depois jogados em outra, tornando-os ideais para jogos de aventura arcade. A tela inicial é alterada com as teclas **Q** e **W**. Uma tela inicial de 255 indica que o objeto inicia o jogo no inventário do jogador. Objetos que estão faltando no início do jogo, ou seja, não aparecem em nenhuma tela ou no inventário do jogador, devem ser atribuídos uma tela inicial de 254. Pressione **P** para colocar um objeto em sua posição designada em sua tela inicial.

**X** = inserir objeto.

**D** = excluir objeto.

**C** = grade de imagem clara.

**M** = copiar imagem ou lado a lado da área de transferência.

**K** = copiar imagem ou bloco da área de transferência

**N** = próximo objeto.

**L** = último objeto.

**Q** = alterar a tela inicial.

**W** = altera a tela inicial.

**P** = posicionar objeto na tela inicial.

Objetos podem ser manipulados no código de eventos com **GOT**, **GET**, **PUT** e **DETECTOBJ**. Além disso, a variável OBJ armazena o resultado do último comando DETECTOBJ, embora você possa usá-lo também para outras coisas, se necessário. **IF GOT n** será verdadeiro se o jogador tiver o objeto **n** em seu inventário. **GET n** colocará o objeto n no inventario do jogador (independentemente de onde o objeto estiver atualmente). **PUT n** derrubará um objeto na posição atual do sprite e DETECTOBJ verificará se o sprite atual está tocando um objeto, colocando o resultado na variável OBI. Se nenhum objeto for detectado na posição do sprite, o valor 255 será retornado.

Por exemplo, o código a seguir selecionará automaticamente qualquer objeto pelo qual um sprite passe:

DETECTOBJ

IF OBJ <> 255

GET OBJ

ENDIF

Como GET e PUT não verificam se o objeto já está no inventário do jogador, eles podem ser usados para colocar objetos na tela ou removê-los mais ou menos à vontade. Você pode desejar que os sprites inimigos abram bônus, depois os remova após alguns segundos. Só porque você remove um objeto da tela não significa que você tenha que conceder os pontos ou bônus que o jogador teria ganhado ao coletá-lo.

Map Layout

AGD permitirá níveis sequenciais, mas também tem a capacidade de criar jogos exploradores onde o jogador pode explorar um mapa.

O "mapa" do seu jogo é organizado como uma grade de 10 x 8 locais, e todos começam vazios. Locais vazios aparecem como dois hifens "-", os quartos aparecem como os números da tela, por exemplo, "01" ou "12". Como cada quarto é projetado no designer de tela, ele pode ser colocado nessa grade em um local escolhido. Para se deslocar pelo mapa, basta utilizar os comandos **SCREENUP**, **SCREENDOWN**, **SCREENLEFT**, **SCREENRIGHT**. Durante o jogo, não será possível ao jogador mover-se para um espaço de grade vazio. Comandos como **NEXTLEVEL** ou **LET SCREEN = 9** não alteram a posição atual do mapa; portanto, são mais bem utilizados em jogos com níveis sequenciais, a menos que você esteja confiante no que está fazendo.

O cursor vermelho pode ser movido ao redor da grade usando as teclas de cursor, para mudar a sala em um determinado local da grade mova as teclas '**1**' e '**2**'. Os quartos são exibidos nos dois terços inferiores da tela conforme são selecionados para facilitar o processo.

Seu mapa seguirá as regras da geometria euclidiana, mas você pode dobrar as regras para criar um campo de jogo distorcido, caso deseje. Mover-se para a esquerda a partir de uma sala situada na borda esquerda do mapa fará com que o jogador reapareça na sala colocada na borda direita da próxima linha, se alguém estiver lá. Da mesma forma, mover-se diretamente de uma sala à direita levará o jogador para a sala na extremidade esquerda da sala, uma linha abaixo. Se você não quiser que isso aconteça, você deve construir paredes nas telas relevantes para formar uma barreira física. Também é possível reutilizar uma sala, para que ela apareça mais de uma vez em seu mapa.

Mover-se de uma tela para outra não alterará as coordenadas do sprite do jogador, portanto, elas devem ser definidas manualmente, ao mesmo tempo em que as opções screenleft, screenright etc. são executadas. Qualquer sprite de jogador configurado para uma tela será usado apenas na primeira tela, ou para gerar um sprite de novo jogador caso ele morra na tela. Se o jogador morrer em uma tela onde ele não tenha uma posição padrão, ele não será reaberto.

Pressione '**X**' para declarar a localização da grade como o ponto no qual o jogador deve começar o jogo.

**1** = selecionar sala anterior da lista.

**2** = selecione a próxima sala da lista.

**X** = selecione a sala onde o jogador começa o jogo.

**ENTER** = retornar ao menu principal.

Jump Table

Se o seu jogo faz uso da gravidade, você pode querer editar a tabela de salto. Isso permite que você edite a inclinação de saltos e / ou quedas, permitindo que você determine como os sprites altos podem pular, ou com que rapidez eles descem quando caem através de falhas no chão.

A tabela de salto é separada em uma série de etapas individuais e você pode alterar a distância entre essas etapas com as teclas do cursor. A coluna vermelha representa a etapa que você está editando no momento. Quando estiver satisfeito com sua tabela de saltos, pressione **ENTER** para retornar ao menu principal.

Sound

**H** = ouve som presente.

**N** = som seguinte.

**P** = som anterior.

**X** = criar novo som.

**D** = apagar som.

Movimente os valores com as teclas do cursor, aumentando e diminuindo-os com **1** e **2**.

Use **3** e **4** para aumentar e diminuir o tom em incrementos de 50. **SPACE** ou **0** desativará o ruído ou o tom.

Para reproduzir um som no seu jogo, use o comando **SOUND** no evento relevante.

Os sons são reproduzidos usando o chip AY. Para efeitos de bipe de 48k, use o comando **BEEP**.

Save Game

Solicita um nome de arquivo para o seu jogo e, em seguida, pergunta se você deseja iniciar o código com um programa de carregador BASIC que carrega e executa automaticamente o seu jogo (e carregando a tela se você configurou um no menu miscelânea), caso deseje execute-o independentemente do utilitário. O jogo é salvo em fita com ou sem o carregador BASIC. Se você salvar o jogo com um carregador, ele retornará ao início quando terminar, caso contrário, retornará ao BASIC para que você possa escrever suas próprias rotinas de final de jogo.

O arquivo de código salvo pode ser carregado posteriormente e editado. Você pode até querer escrever seu próprio carregador BASIC, o programa mais simples para fazer isso seria:

10 CLEAR 31231: LOAD ""CODE: RANDOMIZE USR 32000

You would save this BASIC program to tape with something like

SAVE "MyGame" LINE 10

Claro, você pode querer configurar algumas outras coisas primeiro, como as cores BORDER, PAPER e INK, além de uma página de título e talvez até mesmo uma borda ao redor do painel de status.

Load Game

Carrega um novo jogo da fita. Se você estiver usando um emulador, terá que operar o navegador de fitas por conta própria. Emuladores sem navegador de fita e aqueles que carregam automaticamente as fitas inseridas não são recomendados para o AGD. O AGD só carregará arquivos de código que foram criados com o utilitário.

Test Game

Permite que você teste sua criação. Pressione **ENTER** em qualquer ponto para retornar ao editor.

Miscellaneous

Fornece mais opções para modificar o mecanismo de jogo e mostra o endereço do primeiro local de memória não utilizado disponível, caso deseje marcar suas próprias rotinas no final do jogo.

**Collision Distance**

A distância de colisão é o ajustador de detecção de colisão de sprite. Sprites padrão são 16x16 pixels, então normalmente eles vão se acertar se suas coordenadas estiverem separadas por menos de 16 pixels. Para um sprite robusto como Monty Mole ou Egghead, este é o ideal. No entanto, você pode querer desenhar um sprite jogador que é um pouco mais magro, deixando espaços em torno de ambos os lados. Você não vai querer contar a diferença como parte da detecção de colisão de sprite, então você pode reduzir essa distância para uma contagem menor de pixels. Lembre-se de que essa configuração afeta apenas as distâncias horizontais e define todas as detecções de colisão entre os sprites.

**Sprite Height**

Enquanto todos os sprites devem ter o mesmo tamanho, agora você tem a opção de 16x16 ou 16x24 para o seu jogo. No entanto, esteja ciente de que sprites 16x24 ocupam 50% mais memória. Você será solicitado a confirmar que deseja alterar o tamanho do sprite e, em seguida, o AGD converterá o mecanismo, permitindo a memória. Quaisquer sprites existentes são convertidos para o novo tamanho. Você deve editá-los manualmente no editor de sprites.

**SPRITEINK Mask**

O comando SPRITEINK normalmente permite que você altere a cor INK de um sprite sem afetar sua configuração BRIGHT ou FLASH. Se você quer que o SPRITEINK faça isso, ou está projetando o seu jogo com o ULAplus em mente, você precisará mudar esta configuração. Um valor de 7 muda a INK, 71 muda de INK e BRIGHT, 135 muda de INK e FLASH e 199 muda de INK, BRIGHT e FLASH. Para o ULAplus, um valor de 199 permite que você selecione qualquer uma das quatro tabelas de consulta de cores. Para ligar o BRIGHT, adicione 64 ao valor SPRITEINK da mesma forma que faria com a função BASIC ATTR. Para ativar o FLASH, adicione 128.

**Loading Screen**

Esse recurso permite que você importe uma tela de carregamento criada com outro utilitário. AGD primeiro solicita que você confirme que deseja carregar uma tela de carregamento e, em seguida, carrega uma tela.

**Order Sprites**

Quando os sprites de ordem são ativados, a tabela de sprites é ordenada e a ordem na qual os sprites são desenhados depende de sua posição vertical para eliminar o flicker. Quando isso é desativado, a ordem em que os sprites são desenhados não varia de um quadro para outro.

**Specialise**

Esta opção seleciona qual das três adaptações de especialista você deseja usar para o seu jogo. As adaptações disponíveis são partículas, efeitos e quebra-cabeças. O mecanismo de partículas permite comandos TRAIL, EXPLODE e LASER. É ótimo para trilhas de vapor / poeira mágica, explosões, lasers e, geralmente, para dar aos seus shoot-em-ups um pouco mais de polimento. Efeitos dá aos comandos FADE e TICKER a atenuação da janela e da rolagem de pixel em uma mensagem em um ponto especificado na tela - útil para rolar mensagens em suas intro ou atualizações de mensagens de texto no jogo. Quebra-cabeça habilita o comando INV permitindo que o jogador selecione itens em seu inventário e habilita blocos de água. Ele também lembra de quaisquer blocos posicionados em uma tela usando o comando PUTBLOCK e redesenha-os quando o player sai e, em seguida, entra novamente na tela.

**AY Sounds**

Esta opção pode ser usada para desabilitar os efeitos sonoros AY, caso deseje configurar uma rotina de música AY com interrupções.

Messages

É aqui que você pode definir as mensagens de texto que seu jogo usará. Pressione **ENTER** a qualquer momento para retornar ao menu principal.

**N** = próxima mensagem.

**P** = mensagem anterior.

**SPACE** ou **0** = editar mensagem atual.

**X** = criar nova mensagem.

**D** = excluir mensagem.

Events

Essa é a parte em que você pode falar sobre a lógica do jogo e mudar a maneira como ela funciona de várias formas diferentes. Embora o editor e o compilador não rivalizem com uma linguagem apropriada como o BASIC, o AGD fornece um número limitado de declarações, funções e variáveis que devem permitir a criação de uma variedade de diferentes jogos de arcada ou aventuras de arcada. Pense nisso como uma versão arcade do GAC da Incentive Software - você pode precisar ser criativo sobre como implementar os recursos que deseja, mas isso é metade da diversão.

Existem vários eventos para os quais você pode escrever a lógica. Além daqueles que ocorrem em determinados momentos do jogo, há eventos associados a 8 tipos de sprites. Estes são os eventos que controlam o movimento e a lógica de cada tipo de sprite. Sprite tipo 0 é geralmente reservado para o sprite do jogador, então este código deve testar as chaves e mover o sprite de acordo. O resto é todo seu para fazer o que quiser. Por exemplo, você poderia escolher fazer sprites tipos 1 e 2 diferentes alienígenas com diferentes padrões de movimento, e talvez usar sprite tipo 3 para sprites de bônus que o jogador pega.

Controle do jogador (tipo 0) - movimento do jogador, leitura de chaves, detecção de colisão

Sprite tipo 1 - comportamento de sprites com tipo 1

Sprite tipo 2 - comportamento de sprites com tipo 2

Sprite tipo 3 - comportamento de sprites com tipo 3

Sprite tipo 4 - comportamento de sprites com tipo 4

Sprite tipo 5 - comportamento de sprites com tipo 5

Sprite tipo 6 - comportamento de sprites com tipo 6

Sprite tipo 7 - comportamento de sprites com tipo 7

Sprite tipo 8 - comportamento de sprites com tipo 8

Inicializar sprite - sempre que um sprite é inicializado

Loop principal 1 - todo loop de jogo

Loop principal 2 - todo loop de jogo

Intro /menu - antes do jogo começar

Iniciar o jogo - no início do jogo, configure as variáveis

Reinicie a tela - o que acontece quando o jogador reinicia uma tela

Caiu muito longe - quando qualquer sprite chega ao final da mesa de pulo

Kill player - acontece quando o jogador perde uma vida com um comando KILL

Jogo perdido - quando o jogador perde sua última vida

Jogo concluído - quando o jogo foi concluído com sucesso, por exemplo, parabéns

Para modificar os eventos, use o cursor para cima / baixo para selecionar o evento e, em seguida, pressione espaço ou 0. Se nenhum código tiver sido escrito para esse evento, você será apresentado a um menu adicional de modelos básicos de código para iniciá-lo. Você não precisa usar um modelo, esses são apenas exemplos pré-escritos de código para executar diferentes funções. Quando você seleciona um modelo, o editor copia um pedaço do código no evento para você, mas não é necessário selecionar um.

As opções são:

None

Manic Miner style controls

Player escadas e níveis

Cybernoid style controls

Patrulhando o inimigo (L / R)

Patrulhando o inimigo (U / D)

Inimigo saltitante

Escadas e níveis perseguidores

Static collectable

Bloco empurrável

Plataforma móvel

Minigame Bonus Stopper

A primeira opção pula os modelos e permite que você comece com um pedaço de papel em branco.

As próximas opções são os controles do jogador. Os controles de estilo Manic Miner lhe dão as teclas esquerda, direita e pular. As escadas e níveis dos jogadores usam escadas esquerda, direita e para cima e para baixo. Controles de estilo Cybernoid são deixados, à direita, fogo (apenas motor de partículas) e para cima. Quando não estiver subindo, o sprite do jogador se moverá automaticamente, como em Cybernoid.

Existem modelos de sprites inimigos também. A opção de inimigo de patrulhamento (L / R) se moverá para a esquerda e para a direita ao longo de uma plataforma, revertendo quando atingir uma parede ou se houver uma lacuna no chão. Patrulhar o inimigo (U / D) sobe e desce, revertendo sempre que atinge uma parede, os inimigos quicam apenas saltam diagonalmente ao redor da tela até que eles atinjam algo, então mudam de direção. Escadas e níveis perseguidores caminham ao longo de plataformas e paredes, subindo e descendo escadas.

Itens colecionáveis estáticos são sprites que ficam na tela e esperam por uma colisão com o jogador, e então eles se removem e adicionam à pontuação do jogador, os blocos que podem ser empurrados reagirão às colisões com o sprite do jogador, movendo-se para fora do caminho. Eles também testam colisões com outros sprites do tipo 1 e destroem qualquer um que encontrarem - você pode desejar modificar esse comportamento. As plataformas móveis impedem que os sprites dos jogadores caiam da mesma maneira que uma plataforma padrão faria. O modelo os move para cima e para baixo, mas com uma pequena modificação eles podem ser movidos para a esquerda e para a direita.

Quando um modelo é selecionado ou você seleciona um evento que já possui algum código, você pode editar o código desse evento especifico. Deslocamento de símbolo e A retorna para a tela de seleção de eventos, o editor irá executar rapidamente o seu código e informar se não entender nada, posicionando o cursor na linha que ele não entende, o código é então compilado diretamente para um código de máquina extremamente rápido.

Chaves do editor de código:

Linha de corte SYM-Y

Linha de pasta SYM-U

Página inicial SYM-Q

Fim SYM-E

CAPS-2 Toggle caps lock

CAPS-3 Excluir encaminhar

CAPS-4 Inserir / sobrescrever

SYM-CAPS Alternar modo extendido

Funções só podem ser usadas após um IF.

**IF**

Teste. Se a seguinte condição for verdadeira, o código até a próxima instrução ENDIF será executado. IF pode ser usado com uma função, ou para testar variáveis ou parâmetros de sprites entre si, ou contra valores numéricos específicos.

**ENDIF**

Marca o final do código condicional.

**LET**

Como no BASIC, isso permite atribuir um valor a uma variável ou parâmetro de sprite.

O valor atribuído pode ser um número ou outra variável ou parâmetro de sprite.

**KEY**

Função. Espera um único argumento numérico e a condição é verdadeira se a tecla for pressionada.

**CANGOUP, CANGODOWN, CANGOLEFT, CANGORIGHT**

Funções. A condição é verdadeira se o sprite atual puder se mover para cima / baixo / esquerda / direita.

**LADDERABOVE, LADDERBELOW**

Funções. A condição é verdadeira se o sprite atual puder subir / descer uma escada.

**X, Y**

Parâmetros Sprite. Estas são as coordenadas do sprite atual.

**A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T**

Variáveis globais. Eles contêm valores de 8 bits, portanto, valores de 0 a 255 são possíveis.

**SCREEN, LIVES**

Variáveis globais. Eles contêm o número da tela atual e as vidas restantes.

**TYPE**

Parâmetro Sprite. Este é o tipo de sprite sendo processado. Melhor usado em conjunto com o parâmetro IMAGE, definir este parâmetro irá mudar completamente o comportamento do sprite - útil para transformar um desagradável em um bônus, ou fazer um sprite parar e explodir antes de matá-lo. Não há razão para que você não possa mudar um sprite para digitar zero e colocá-lo sob o controle do jogador, ou mudar o tipo de sprite do jogador para outra coisa com um conjunto de controles ligeiramente diferente. Desde que o novo tipo de sprite tenha o código apropriado configurado no evento relevante, não há limite para o que você poderia fazer. Se você não tiver configurado nenhum código para o sprite, seu sprite ficará lá - o que pode ser bom se você quiser.

**IMAGE, FRAME**

Parâmetros Sprite. Estes são os números de sprite e frame mostrados no editor de sprite. Você pode mudar o sprite de acordo com a direção que o jogador estiver enfrentando, ou talvez você queira dar ao jogador uma escolha de veículos para controlar. Definir um número de quadros além do limite do sprite resultará em uma imagem de sprite diferente sendo exibida, então use FRAME com cuidado. Ao definir a imagem, é uma boa ideia definir o quadro como **0** ao mesmo tempo, a menos que você tenha uma boa razão para não fazê-lo.

**DIRECTION, PARAMA, PARAMB**

Parâmetros Sprite. Você pode usá-las como quiser, talvez para indicar a direção na qual um sprite em particular está se movendo ou a fase especifica pela qual está passando. Inestimável para qualquer forma de IA inimiga.

**ANIMATE**

Comando. Anima o sprite atual, percorrendo automaticamente os quadros em ordem crescente.

**ANIMBACK**

Comando. Como ANIMATE, mas percorre os quadros em ordem decrescente.

**NEXTLEVEL, RESTART**

Comando. Mover para o próximo nível e reiniciar o nível atual, respectivamente.

**SPRITEUP, SPRITEDOWN, SPRITELEFT, SPRITERIGHT**

Comandos Mova o sprite atual de acordo. Nenhuma verificação é feita para blocos no caminho, ou condições fora da tela, então você terá que fazer isso com funções como CANGOLEFT ou LADDERABOVE.

**SPAWN**

Comando. Espera 2 parâmetros para o tipo e imagem do sprite. Isso gera um novo sprite com o tipo e imagem especificados na posição dos sprites atuais. O novo sprite é criado com FRAME, DIRECTION, PARAMA e PARAMB todos configurados para zero. O sprite atual não é afetado.

**SPAWNED**

Comando. Deve ser usado somente após uma instrução SPAWN. Este comando seleciona o sprite recém gerado. Qualquer código escrito após SPAWNED se referirá ao novo sprite. Use ORIGINAL para voltar ao sprite primário original. Alternativamente, você pode preferir colocar seu código no evento inicialização do sprite, depende de você.

**REMOVE**

Comando. Remove o sprite atual da tabela. Útil para destruir inimigos ou pegar objetos.

**PUTBLOCK**

Comando. Requer um parâmetro para especificar o número do bloco, isso coloca o bloco especificado na posição atual de LINE e COLUMN na tela e pode ser útil para abrir e fechar portas. Note que sair e voltar a entrar no ecrã ou redesenhar depois de MENU ou INV, irá restaurar o ecrã para a sua posição predefinida, a menos que tenha selecionado a especialização de quebra-cabeças.

**COLLISION**

Função. Requer um argumento numérico para especificar o tipo de sprite. A condição é verdadeira se o sprite atual estiver em colisão com outro sprite do tipo especificado.

**OTHER**

Comando. Deve ser usado somente após uma verificação de COLISÃO bem-sucedida. Este comando seleciona o outro sprite, isto é, o secundário com o qual o sprite original acabou de colidir. Qualquer código escrito depois de OTHER se referirá ao sprite secundário. Use ORIGINAL para voltar ao sprite primário original quando terminar o código para o sprite secundário.

**ORIGINAL**

Comando, usado após os comandos OTHER e SPAWNED, este reverte para o sprite original.

**ENDGAME**

Comando. Termina o jogo em vitória. Isso realiza o evento Completed Game. Completar a última tela em um jogo com níveis sequenciais fará o mesmo. A única outra maneira em que um jogo pode terminar é se o jogador perder todas as suas vidas, mas isso não executa o evento Completed Game.

**SHOWSCORE**

Comando. Mostra a pontuação na posição atual do cursor. Deve ser imediatamente precedido por instruções para configurar a linha e a posição da coluna para o cursor.

**SCORE**

Comando. Espera ser seguido por um número para adicionar à pontuação. SCORE 100 adicionará 100 pontos ao total do jogador. Os valores de 0 a 255 são válidos.

**SOUND**

Comando. Inicia um efeito sonoro com o chip de som AY. Espera um único parâmetro para o número de som para jogar.

**BEEP**

Comando. Inicia um efeito sonoro do bip. Espera um único parâmetro para a duração do som. Valores acima de 127 produzem efeitos de ruído branco.

**CLS**

Comando. Limpa a tela.

**BORDER**

Comando. Espera um argumento de 0 a 7 inclusive. Define a cor da borda.

**COLOUR**

Comando. Espera um argumento. Define os atributos de exibição permanentes, o formato é 128 \* FLASH + 64 \* BRIGHT + 8 \* PAPER + INK. Pode ser usado antes de exibir texto ou limpar a tela.

**DELAY**

Comando. Espera um argumento. Pausa pela duração especificada.

**MESSAGE**

Comando. Espera um argumento para o número da mensagem a ser exibida.

**KILL**

Comando. Inicia o evento kill player e diminui o contador de vida. Você deve configurar o contador de vidas no evento de inicialização usando algo como LET LIVES = 3.

**LINE, COLUMN**

Variáveis. Estes determinam a posição em que a pontuação ou mensagem será exibida. Eles são temporários e mudam toda vez que um sprite é exibido, portanto, sempre os use imediatamente antes de um comando MESSAGE ou SHOWSCORE.

**GETRANDOM**

Função. Gera um número aleatório entre zero e o argumento e coloca-o na variável RND. GETRANDOM 100 gerará um número de 0 a 99, GETRANDOM 2 gerará um zero ou 1.

**RND**

Variável. O último número aleatório gerado pelo GETRANDOM.

**ADD, SUBTRACT**

Comandos. Adicione ou subtraia de ou para um parâmetro ou variável de sprite, por exemplo: ADD 1 TO A ou SUBTRACT 5 FROM B.

**DISPLAY**

Comando. Exibe o valor numérico na variável ou parâmetro a seguir. Por exemplo,

DISPLAY LIVES.

**SCREENUP, SCREENDOWN, SCREENRIGHT, SCREENLEFT**

Comandos. Mover para cima, para baixo, esquerda ou direita, uma tela, se possível.

**DEADLY**

Função. A condição é verdadeira se o sprite atual estiver em contato com um bloco mortal.

**CUSTOM**

Função. Semelhante a DEADLY, true se o sprite atual estiver em contato com um bloco CUSTOM.

**WAITKEY**

Comando. Espera por um pressionamento de tecla.

**JUMP**

Comando. O sprite atual irá pular, desde que não esteja no ar e não haja paredes no caminho. Como você esperaria, qualquer sprite pode ser feito para pular, não apenas aqueles sob o controle direto do jogador.

**FALL**

Comando. Desde que o sprite já não esteja caindo ou pulando, isso verifica se o sprite está em pé em cima do solo sólido. Se não, ele começará a cair, usando a metade descendente da mesa de pulo (seu jogo encontrará automaticamente este ponto, mesmo que você tenha modificado a mesa de pulo). A QUEDA(FAAL) é útil se um tipo de sprite estiver sujeito à gravidade, por exemplo, jogos de plataforma.

**GOT**

Função, usado em instruções IF e espera um único argumento. A expressão é verdadeira se o objeto especificado for de propriedade do player. O argumento pode ser uma variável se necessário, então IF GOT 1 é válido, assim como IF GOT A. Você pode até usar um parâmetro sprite como um argumento, se desejar.

**GET**

Comando. Espera um parâmetro especificando o objeto a ser obtido. Coloca o objeto especificado no inventário do jogador, independentemente de onde ele esteja. O objeto pode estar na tela atual, em outra tela ou simplesmente ausente. Cabe a você decidir quando o jogador pode obter um objeto especifico. Você pode querer usar DETECTOBJECT para determinar quando um sprite está tocando um objeto primeiro.

**PUT**

Comando. Requer um argumento especificando o número do objeto. O objeto é solto na tela na posição do sprite atual, a menos que já esteja na tela atual. O argumento pode ser numérico, um parâmetro de variável ou sprite.

**DETECTOBJ**

Comando. Detecta objetos tocados pelo sprite atual e coloca o resultado na variável OBJ. Se o sprite estiver tocando em mais de um objeto, somente o objeto com o menor número será detectado. Se nenhum objeto for detectado, o OBJ será configurado para 255.

**OBJ**

Variável. O último número do objeto detectado pelo DETECTOBJ.

**CLOCK**

Variável de leitura. O contador de quadros atual, em 50 segundos. Útil para horários.

**MENU**

Comando. Requer um argumento especificando o número da mensagem. Produz um menu pop-up no meio da área de reprodução, onde as opções são tiradas do número da mensagem especificada, cada uma em sua própria linha. O jogador então seleciona uma opção usando os controles para cima, para baixo e fogo. Geralmente isso acontece quando o jogador pressiona uma determinada tecla. No entanto, ele pode aparecer em outro evento de sprite, por exemplo, quando um determinado sprite colide com o jogador. Você pode exibir uma mensagem - a conversa do sprite com o jogador - e usar o MENU para solicitar uma resposta. A tela é redesenhada após o MENU.

**INV**

Comando. Requer um argumento especificando o número da mensagem, semelhante ao menu, mas isso produz um menu baseado em objetos em posse do jogador. Configure uma lista dos nomes de objetos usados em seu jogo em uma mensagem com cada item em uma linha separada e use INV n, onde n é o número da mensagem que contém os nomes dos objetos. INV apresentará ao jogador um menu contendo apenas os itens atualmente no inventário do jogador. Útil para selecionar um item para processar ou descartar. INV trabalha apenas com a especialização de quebra-cabeças.

**EXIT**

Comando. Todo o processamento do evento atual é finalizado.

**REPEAT**

Comando. Requer um parâmetro para determinar o número de repetições. O código até o próximo ENDREPEAT será repetido o número de vezes especificado. REPEAT não pode ser aninhado.

**ENDREPEAT**

Comando. Isso marca o fim do bloco de código a ser repetido. O código colocado em um loop REPEAT..ENDREPEAT é automaticamente recuado pelo editor para melhorar a legibilidade.

**MULTIPLY**

Comando. Multiplique um parâmetro ou variável de sprite, por exemplo.

MULTIPLICAR C (BY)POR 7.

**DIVIDE**

Comando. Divida um parâmetro ou variável de sprite, por exemplo. DIVIDE Y POR 8.

**SPRITEINK**

Comando. Define a cor da tinta para o sprite atual para a cor especificada, por exemplo, SPRITEINK 4 OU SPRITEINK PARAMB. Não afeta PAPER, FLASH OU BRIGHT. Para evitar deixar um rastro de INK com cores diferentes, redefina o SPRITEINK no início de cada evento de sprite. Se o seu fundo INK é normalmente 7 (branco), você pode querer usar o SPRITEINK 7 no começo do evento sprite, mover o sprite normalmente, então configure para a cor que você quiser. Os sprites colecionáveis estáticos só devem redefinir o SPRITEINK para o INK de fundo normal quando forem coletados. Veja o jogo de demonstração Diamond Geezer por um exemplo.

**LASER**

Comando. Espera um argumento numérico. Dispara um laser para a esquerda ou para a direita a partir da posição atual do sprite, dependendo do parâmetro. Até mesmo os parâmetros são disparados, os impares disparam à direita. As colisões de sprites com lasers podem ser detectadas nos eventos do tipo sprite usando IF COLLISION 10. LASER só funciona se a especialização de partículas for selecionada no menu diversos.

**TRAIL**

Comando. Útil para poeira de fada ou trilhas de vapor, TRAIL cria uma partícula de trilha de vapor no centro do sprite atual. Usando ADD 8 TO Y ou SUBSTRACT 8 FROM Y primeiro, você pode criar uma trilha para a esquerda ou para a direita do sprite. Lembre-se de adicionar ou subtrair 8 novamente apôs o comando TRAIL para restaurar as coordenadas originais do sprite. Só trabalha com a especialização de partículas.

**EXPLODE**

Comando. Requer um argumento numérico para especificar o número de partículas. Cria uma explosão na posição atual do sprite. Usado com REMOVE, o EXPLODE remove a necessidade de desenhar uma animação de explosão para seus sprites. Se você esquecer de seguir EXPLODE com um parâmetro, o próximo comando não será compilado corretamente. Se esse comando for REMOVE seu sprite não será removido e você vai se perguntar por que explode, mas continua em jogo. Especialização de partículas apenas.

**FADE**

Comando. Transforma lentamente a janela para preto. O Fade só funciona com a especialização de efeitos ativada.

**TICKER**

Comando. Pixel rola uma mensagem de texto na posição atual LINE e COLUMN. Espera 2 argumentos, o primeiro é o número da mensagem e o segundo a largura da mensagem em caracteres. O TICKER continua e repete a mensagem quando chega ao fim. TICKER 1 16 rolará a mensagem 1 em uma faixa com metade da largura da tela. Para desativar TICKER, especifique um número de mensagem que não existe, por exemplo, TICKER 99 1. Só funciona se a especialização de efeitos estiver ativada.

**BIGMESSAGE**

Comando. Requer um parâmetro especificando uma mensagem de texto. Exibe a mensagem na tela em texto de altura dupla. Apenas inicialização de efeitos.

**REDRAW**

Comando. Redesenha a tela, sprites, objetos e quaisquer partículas. Se você está escrevendo uma aventura arcade, você pode querer exibir mensagens na janela do jogo, pausar por uma tecla com WAITKEY e depois REDRAW antes de retornar à ação.

**SILENCE**

Comando, silencia o chip de som AY.

**CONTROLMENU**

Comando. Útil para o evento de menu / intro, o CONTROLMENU espera que o jogador selecione uma opção de controle. As opções são 1 para teclado, 2 para joystick Kempston e 3 para joystick Sinclair (teclas 67890).

**DIG**

Comando. Espera um parâmetro de 0 a 3, especificando a direção para cima, para baixo, para a esquerda ou para a direita, respectivamente. Remove os blocos do FODDER imediatamente adjacentes ao sprite do jogador. Disponível apenas com a especialização em quebra-cabeças.

**STAR**

Comando. Parte do motor de partículas, STAR cria um campo de estrelas vertical ou horizontal quando chamado repetidamente. Espera um parâmetro para especificar a direção, 0 para vertical e 1 para horizontal.

**BIGSCORE**

Comando. Como SHOWSCORE, mas exibe a mensagem na tela em texto de altura dupla. Apenas inicialização de efeitos.

**STOPFALL**

Comando. Pára um sprite pulando ou caindo no ar e invoca o evento caiu muito longe, se aplicável.

**Exemplo 1**

Suponha que temos uma janela de jogo de 22 caracteres de altura com um único espaço entre os caracteres na parte superior e inferior da tela. Digamos que queremos desenhar uma caixa ao redor da janela. Podemos fazer isso exibindo mensagens no evento de inicialização do jogo. Poderíamos configurar a mensagem 0 como a linha superior a ser exibida acima da janela, com a mensagem 1 como a linha inferior. Para desenhar as linhas nas laterais, podemos configurá-las na mensagem 2 e, em seguida, usar um loop REPEAT para desenhá-las. Algo assim pode fazer o truque para nós:

LET LINE = 0

MESSAGE 0

REPEAT 22

MESSAGE 2

ENDREPEAT

MESSAGE 1

**Exemplo 2**

Se selecionarmos sprite type 3 e escolher Static collectable como nosso template, serão apresentadas as seguintes linhas de código:

IF COLLISION 0

REMOVE

SCORE 100

ENDIF

Podemos adicionar um efeito sonoro de bipe curto inserindo o comando BEEP 30:

IF COLLISION 0

BEEP 30

REMOVE

SCORE 100

ENDIF

Indo mais além, podemos inserir as seguintes linhas no começo, podemos mudar o comportamento do sprite:

IF CANGOUP

SPRITEUP

ELSE

REMOVE

ENDIF

Então nosso evento agora é assim:

IF CANGOUP

SPRITEUP

ELSE

REMOVE

ENDIF

IF COLLISION 0

BEEP 30

REMOVE

SCORE 100

ENDIF

Agora temos um sprite colecionável que sobe até atingir algo e depois desaparece como uma bolha. Agora precisamos de algum código para gerar nossos novos sprites na parte interior da teia. Podemos fazer isso no loop principal, então selecione o evento Main loop 1 e insira estas linhas:

GETRANDOM 127

IF RND <= 6

SPAWN 3 0

SPAWNED

IF TYPE = 3

LET X = 168

GETRANDOM 160

LET Y = RND

ADD 8 TO Y

ENDIF

ORIGINAL

ENDIF

Este código primeiro gera um número aleatório de 0 a 127, e se esse número for menor ou igual a 6, ele gera um sprite do tipo 3 (o evento anterior que editamos) e uma imagem de sprite de 0. Se o sprite tiver gerado corretamente, ele define a posição x para a parte inferior da área de reprodução (supondo que a área de reprodução termine 8 pixels a partir da parte inferior da tela) e escolhe uma posição y aleatória entre 8 e 168.

**Exemplo 3**

Suponha que estamos escrevendo um jogo de labirinto de cima para baixo no estilo Atic Atac e queremos incluir o patrulhamento de inimigos. Podemos selecionar o tipo de sprite 4 e escolher o modelo inimigo de patrulhamento (L / R). No entanto, este modelo pressupõe que estamos escrevendo um jogo de plataforma, portanto, precisaremos remover as linhas que verificam as lacunas, pois o sprite não estará em uma plataforma. No modelo, essa verificação é feita adicionando ou subtraindo 16 (a largura do sprite) à coordenada horizontal (Y) do sprite e testando para ver se o sprite pode ficar inativo. Se puder, o sprite muda de direção porque deve haver uma lacuna. No nosso caso, nós simplesmente queremos nos mover para a esquerda ou para a direita até atingirmos alguma coisa, então nos viramos, então é uma questão de deletar as linhas desnecessárias. Nosso modelo deve ficar assim:

IF PARAMA = 0

IF CANGOLEFT

SPRITELEFT

ELSE

LET PARAMA = 1

ENDIF

ELSE

IF CANGORIGHT

SPRITERIGHT

ELSE

LET PARAMA = 0

ENDIF

ENDIF

Character Set

O AGD vem com seu próprio editor de fontes. Os caracteres são editados usando as teclas do cursor com **0** ou **SPACE** para definir / desmarcar um pixel. Todo o intervalo imprimível de 96 caracteres é exibido na parte inferior da tela, e você pode percorrê-los com as teclas **N** e **P**.

Se preferir, você pode carregar uma fonte padrão do Spectrum criada com outro utilitário no AGD. Primeiro, certifique-se de que, se estiver usando um emulador, ele não carregará automaticamente as fitas ou perderá o jogo em que você está trabalhando. Pressione **L** para carregar uma fonte.

**P** = caractere anterior

**N** = caractere seguinte

**L** = carregar fonte da fita

**ENTER** = retornar ao menu principal

Informação técnica

Os jogos criados com o utilitário são independentes e devem funcionar independentemente do próprio editor. AGD não altera o modo de interrupção, então você está livre para configurar suas próprias interrupções, para tocar música, por exemplo. No entanto, você não deve desabilitar as interrupções. Se você deseja configurar suas próprias interrupções usando o IM2, certifique-se de que sua rotina de serviço incrementa o byte no local da memória 23672 a cada quadro, ou seu jogo irá travar. AGD não altera o valor do registrador IY, embora IX seja usado em todo lugar, principal mente como um ponteiro de sprite.

Os jogos ocupam a memória de cerca de 31200 para cima, e esta é a área que é salva pelo utilitário, o endereço inicial exato varia conforme os eventos não críticos crescem de 31232 para criar espaço na memória RAM superior do jogo. Abaixo disso, o código para o editor está localizado, sendo trocado conforme exigido de outros bancos de RAM. No topo da RAM, os últimos 768 bytes de 64768 a 65535 são usados como uma área de mapa de colisões para distinguir entre diferentes tipos de blocos - paredes, escadas, espaço vazio e assim por diante. No final do jogo há uma área de 300 bytes usada para o motor de partículas caso você decida usar lasers, trilhas de vapor ou explosões. Esse buffer não existe nos mecanismos de quebra-cabeças ou efeitos. Nenhum buffer é necessário para a descompactação de dados de tela, uma vez que eles são expandidos dinamicamente pela rotina que desenha a tela.

Ao salvar suas criações, a opção de salvar um carregador BASIC criará automaticamente uma versão de carregamento automático do seu jogo, completa com uma tela de carregamento, se você configurá-la no menu diverso. Os jogos de execução automática não retornam ao BASIC, fazendo loop continuamente. Portanto, você precisará programar sua própria tela de introdução na AGD. Se você optar por não criar um carregador BASIC, seu jogo retornará ao BASIC no final.

Os jogos podem ser controlados via teclado ou joystick Kempston, e isso é feito via byte no endereço 32005. Um valor de 1 significa que os controles Kempston são usados para as primeiras cinco teclas, 2 usa as teclas Sinclair 6, 7, 8, 9 e zero. Qualquer outro valor é padronizado para os controles do teclado.

O AGD define a variável de sistema de taxa de repetição do teclado no endereço 23562. A ROM do Sinclair define isso como 5 na inicialização e a AGD reduzirá isso para 3 para acelerar a edição. No entanto, se você POKE 23562 com qualquer valor diferente de 5 AGD não irá alterá-lo. Então você pode POKE 23562,1 para definir a taxa de repetição do teclado para a velocidade máxima, caso deseje fazê-lo.

**Adicionando música ou sub-rotinas de terceiros (apenas partícula / efeitos)**

Se você quiser adicionar suas próprias rotinas ao jogo, você deve adicionar estas três linhas temporariamente ao evento onde você deseja chamar a rotina externa:

ASM 205

ASM 85

ASM 5

Não se preocupe muito sobre como isso funciona, é simplesmente uma chamada fictícia para um ponto na ROM Sinclair que não faz nada. Nós vamos mudar isso mais tarde.

Em seguida, selecione diversos no menu e anote o endereço exibido na parte inferior da tela. Este é o seu endereço de compilação, a localização da memória em que você deve compilar sua rotina externa. Em seguida, usando uma calculadora, você precisa calcular dois valores para colocar no(s) evento(s) onde você adicionou anteriormente os valores do ASM. Você precisa calcular seu endereço de compilação dividido por 256 e também o restante.

Volte para o(s) evento(s) onde você coloca os valores ASM, em seguida, substitua 5 pelo endereço de compilação dividido por 256, e 85 com o restante ENTÃO SALVE IMEDIATAMENTE SEU JOGO SEM A OPÇÃO BÁSICA DO CARREGADOR. Não o teste neste ponto, pois o AGD irá travar porque sua rotina não está na memória, você precisará certificar-se de que seu carregador BASIC carrega sua própria rotina além do seu jogo.

**Kempston Mouse**

Poucos jogos do Spectrum fazem uso do controle do mouse, e o AGD não foi projetado especificamente para funcionar com esse dispositivo. No entanto, o AGD tem uma rotina oculta para ler o mouse Kempston. Para usá-lo, selecione a especialização de aventura no menu miscelânea e, em seguida, coloque esses três comandos em um dos seus principais eventos de loop:

ASM 205

ASM 192

ASM 129

Isso colocará as coordenadas do mouse e os status dos botões de disparo nas variáveis A, B e C.

Outras informações

Paul Jenkinson produziu uma série de excelentes tutoriais em vídeo da AGD, que podem ser encontrados em seu site:

<http://www.randomkak.bloaspot.co.uk/p/aad-video-tutoria1s.html>

O fórum oficial da AGD é o lugar para encontrar notícias sobre os últimos lançamentos, dicas sobre como tirar o melhor proveito do utilitário e detalhes de jogos que já foram escritos com a AGD ou estão atualmente em desenvolvimento. Você pode encontrar o fórum aqui:

http://arcadeqamedesioner.proboards.com/

Outros recursos:

<http://www.funspot.it/cateaorv/aad/>

http://www.vintageisthenewold.com/agd-arcade-game-designer-initial-setp

