RetroGameEngine X16

Introduction

With the RetroGameEngine x16, anyone can develop 2D top-down view RPG-style games, and in the near future, platform games like Mario Bross. The engine includes a tile editor, map editor, scripting language, events, timers, and functions for sound and graphics.

Directory structure

The main program expects a subdirectory with the name "assets". In this directory it saves and loafs level files, tiles and tilesets.

Also when using SHOWIMG and PLAYZCM etc, the files you specify should be in this subdirectory.

Deploy a game

If you want to deploy a game, you need to deploy the following files in this structure:

- RGE.PRG (you can rename it to match your games name)
- HELPER.BIN
- /ASSETS
  - Your level files
  - Your music/sound files
  - Your binary bitmaps and bmx files (images)

So tiles and tilesets do not need to be included, as they are stored in the level file.

When deploying, rename your first level to 'game.bin'. This way the engine will load this level when started and starts play-mode, skipping the edit-mode.

Keys, mouse and gamepad

In edit mode, you can use the following interact options:

- ALT:	show marker for tiles with events
- CTRL+LEFTCLICK:	replace all same tiles with selected tile
- SHIFT+LEFTCLICK:	pickup tile to draw with
- LEFTCLICK:	draw selected tile in tilemap
- RIGHTCLICK:	open context menu
- T:	toggle tileset on and off
- P:	toggle page of tileset
- O:	open level
- S:	save level

- Arrows:	scroll
- SHIFT+Arrows:	fast scroll
- F5:	toggle between edit mode and play mode
- ESC:	quit

In play mode:

- Arrows:	move player
- Space:	interact with tile (this key can be changed)
- D-pad:	move player
- Button	A: interact with tile
-	

Tiles

On startup, you'll find an empty tilemap and an empty tileset. You can draw each tile manually by right-clicking on a tile in the tileset (on the right of the screen) and selecting 'Edit tile'. Alternatively, you can load a tile or tileset from file. A tile should be a 256-byte bitmap file, based on the default Vera palette, where each byte represents one pixel.

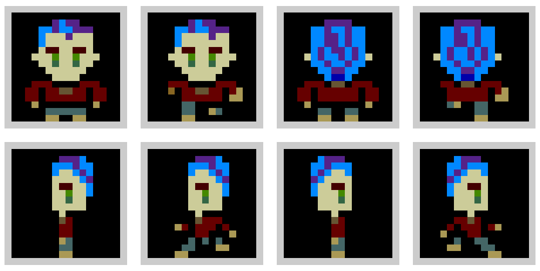
You can also import a tileset from TilemapEd (tiles.bin). This file should have a 2-byte header (which will be ignored), followed by up to 68 tiles of 256 bytes each.

The tileset can contain up to 68 tiles per level. You can toggle the tileset off and on by hitting the T key. Press the P key to swap between the two pages of the tileset. When hovering over a tile in the tileset, the tile number is displayed at the bottom of the tileset.

Player Tiles

The first 8 tiles are reserved for player tiles:

- Tile #0: Front-facing player
- Tile #1: Front-facing walking player
- Tile #2: Backward-facing player
- Tile #3: Backward-facing walking player
- Tile #4: Left-facing player
- Tile #5: Left-facing walking player
- Tile #6: Right-facing player
- Tile #7: Right-facing walking player



The engine will alternate between the two tiles per direction to create a walking animation effect. The transparent color of the player tiles should be color index #0. If you want to add black pixels, use index color #16 instead of #0.

You can set the player's starting position on the map by right-clicking somewhere on the tilemap and selecting 'Set player'.

Special Tile Functions

By right-clicking on a tile in the tileset, you get a context menu with the following options:

• Move up/move down:	Moves a tile up or down in the tileset.
• Edit tile:	Loads the tile in the tile editor.
• Delete tile:	Deletes the tile.
• Transform > Flip H:	Flips the tile horizontally.
• Transform > Flip V:	Flips the tile vertically.
• Transform > Rotate:	Rotates the tile 90 degrees.

Tile Properties and Animations

By right-clicking on a tile in the tileset and selecting ‘Properties’, a dialog opens with the following options, which you can change by pressing the corresponding letter on the keyboard:

- A: Set if the tile is accessible by the player or not.
- B: Iterate through several animation options (scroll horizontal, scroll vertical, animate with next tile(s), rotate).
- C: Sets the starting pixel column or row for the scroll animations.
- D: Sets the ending pixel column or row for the scroll animations.
- E: Sets the animation speed.

If the animation option ‘Next tiles’ is selected, C sets the number of consecutive tiles the engine should use to perform the animations. For example: if you set it to 3, a series of three consecutive tiles (starting from the current tile) will be iterated through, to create an animation effect.

Tilemap

Each level consists of a map of 128x128 tiles. By selecting a tile in the tileset and left-clicking on the map, you can draw the tile on the map.

- CTRL + left-click: Replaces all tiles of the same type under the mouse cursor with the selected tile.
- SHIFT + left-click: Selects the tile under the mouse cursor in the tileset to draw with that tile.

In Edit mode, you can scroll through the tilemap with the arrow keys. By holding down SHIFT while using the arrow keys, scrolling is faster. When hovering over a tile on the tilemap, its x and y coordinates are displayed at the bottom of the tileset.

Saving and Opening Levels

Press S to save the level by entering a filename. To overwrite an existing file (this is not checked!), enter: “@:yourfilename”.
Press O to open a level by entering a filename.

Testing your game

Press F5 to enter play mode, where the screen will be centered on the player sprite. You can then move the player sprite with the cursor keys or a gamepad connected to port #1. To trigger the interact event, press [space] on the keyboard or button A on the gamepad.

Press F5 again to return to edit mode.

Events

The engine supports several events on various levels. If you right-click on a tile in the *tileset*, you can choose from three events:

- On enter: When the player enters a tile or bumps into an inaccessible tile.
- On interact: When the player hits [space] or the A-button.
- On leave: When the player leaves a tile.

When selecting one of these events, an editor window will pop up where you can enter commands implemented by the scripting language. The events you set on a tile in the tileset will be executed for all of these tiles on the tilemap.

If you right-click on a tile on the *tilemap* and choose one of these events, the script will only be executed for that specific tile at that coordinate.

By right-clicking on a tile in the tilemap and selecting ‘Events > On startup’, you get an editor where you can add script commands executed once when the level starts.

Tip:
When in Edit mode you can hold the ALT-key down to visually see which tiles on the tilemap have which events. Each type of event is shown in a different color.

Timers

When you use the SETTIMEOUT command in the script editor, the engine will execute the script in a timer slot after a specified amount of time. You can edit the script of a timer slot by right-clicking on the tilemap and selecting ‘Timer slots’. You are prompted to enter the number of a timer slot to edit its script. You can enter a value between 0 and 127, giving access to 128 slots.

Scripting Language

The scripting language supports a set of commands outlined below. The engine also has registers, which can be viewed as one-byte variables. Registers 0 to 49 are free for your use. Registers 50 and up have specific purposes, which are outlined below.

The script editor is basic, but forgiving and does not have any syntax checking. It's up to you to prevent typing errors. All text that does not correspond to a command will be ignored.

Scripting Commands

TELEPORT x,y	Teleports player to tile coordinates xy.
FADETO n	Fades the screen to a specified intensity level n is a value between 0 (all black) and 15 (normal intensity).
SETTILE (x,y)=n	Sets tile at coordinate x,y to tile number n from the tileset.
SETREG n=”B” SETREG n=123	Sets the internal register n to a new value specified by a char or a value (max 255)
ADDREG r,n	Adds number n to register r.
SBCREG r,n	Subtracts number n from register r.
IFREG n=”B” {…} IFREG n=123 {…}	Conditionally executes code based on the condition
IFTILE (x,y)=n {…}	Conditionally executes code between brackets if the tile number of a specific coordinate is equal to n.
GOTO #label	Resumes execution at specified label. A label should be preceeded with a # Example GOTO #MYLABEL ;This code will be skipped #MYLABEL:
CLEARSCREEN()	clears the screen from all messages etc
EXIT()	stops executing the current event
SHOWMSG “msg”,[position]	displays a message to the users screen. Inline insert of {1} displays the decimal value of register #1. {@1] displays the char of value from register #1 [position] is optional. By default the message will be displayed in the center of the screen. By setting

	position to 1, the message is displayed at the bottom of the screen. Example: SHOWMSG “YOU ARE AT COORDINATE: {60},{61}” As Registers #60 and #61 hold the current coordinates of the player, the message will display its actual coordinates.
GETUSERKEY()	will wait for a keystroke from the user and stores it in Register #50
COPYREG n TO m	copy the contents register n to register m
FLIPV x,y	flips tile x,y in the map verticaly
FLIPH x,y	flips tile x,y in the map horiontaly
SLEEP n	sleeps for n/60 of a second So n=30 is half a second. Max value=255
SETTIMEOUT m,n,[timer slot]	executes a timer slot after a specific amount of time. N=a counter which counts down every 1/60 of a second (0-255) M=When N is 0, M will be decremented by one. (0-255) When M and N reaches zero, the code in timer slot [timer slot] will be executed. Slotnumber can be any value between 0 and 127. Registers 62 to 67 are saved to be used when the timeslot code is being executed. So the values in registers 62 to 67 set when executing SETTIMEOUT will be available at the same register numbers in the timer slot code. This is a way to pass values to the timer slot code.
TILESETPROP n,m=v	sets the property of a tile in the tileset N=tilenumber M=propertynumber: Property numbers: 1=Accessible (0=no, 1=yes) 2=animated (0=no, 1=horizontal,2=vertical,3=animate next tiles,4=rotate) 3=animate px start (1-16) 4=animate px end (1-16)

	5=animate speed (0-16) (0=frozen, 16=slowest, 1=fastest) V=new value If animated=3 (animate with next tiles) PXstart=amount of tiles to use PXEnd must be set to 1
SETPSG channel,freq,waveform,volume	Set a psg channel to output sound Channel: 0-15 Frequency: 0-65550 Wavevorm: 0-3 (0=pulse, 1=sawtooth, 2=triangle,3=noise) Volume: 0-127 Tip: use SETTIMEOUT to set the volume of a channel back to 0 to stop playing after a specific amount of time.
PLAYZCM “filename”,[volume]	Plays a ZCM sound file. Filename: filename of zcm file. Volume: value between 0 and 127 Caution: playing zcm file from disk means disk io is done within an isr, which could cause unexpected behaviour. Use at your risc or use zcm files of max 4kb in size. Alternatively you can use LOADSND and PLAYSND, but then zcm files are loaded into memory before being played.
SHOWIMG “filename”	Load a binary bitmap file, where each byte in the file is an index of the default vera palette. An image of 320x240px is expected.
SHOWBMX x,y,”filename”	Draws an image in BMX format at pixel offset x,y. Caution: a bmx file usually contains a custom palette. This palette is ignored by SHOWBMX. You should create the bmx with the default vera palette, to make sure colors match.
LOADLEVEL “filename”	Load a new level and start playing it. Caution: after the loadlevel command, no more code is being executed!
LOADSND “key”,”filename”,{type}	There is 280kb allocated for music and sound effect files, which can be loaded into memory with LOADSND.

	After being loaded, the sound effects and music can my played with PLAYSND “key”: identifier, max 10 chars example “FIRE” “filename”: filename to load in memory {type}: 0=zcm file, 1=zsm file
PLAYSND “key”,{volume},{loop}	Plays a sound effect or music from memory. Use LOADSND prior to PLAYSND “key”: identifier used with LOADSND {volume}: value between 0-127; 0=silent, 127=max volume {loop}:0=no loop, 1=loop indefinatly
STOPZCM()	Stops playing of ZCM/PCM channel.
STOPZSM()	Stops playing of ZSM music.
REPEAT n .... ENDRPT	Repeat the code between REPEAT and ENDRPT n times. n can be any value between 0 and 255 REPEAT-statements can not nested Register 72 holds the n-value for each repeat sequence. Starting from n and decrementing for each iteration.

Registers

The engine uses registers more or less as variables. The first 50 registers are for you to use as you like, registers 50 and up have special meaning and can be used within the scripting language

0-49	Available to you
50	Last pressed key from GETUSERKEY() function
51	Text for ground color
52	Text background color
53	Background color message window
54	Border color message window
55	Player walking speed (0-255). Lower is faster, default 50.
56	Last walking direction (0 = up, 1 = right, 2 = down, 3 = left)
57	Fade delay: Determines how fast the fading happens. Default: 200 (the lower the value, the faster).
58	Scancode to activate interact event by player. Default= $3D=spacebar. (see https://cx16forum.com/forum/viewtopic.php?t=6559 for scancodes)
60	Current tile X coordinate
61	Current tile Y coordinate
62-67	Parameters to pass to SETTIMEOUT execution

72	Repeat counter for REPEAT statement

Using registers as variables

When specifying a number like SETTILE (5,6)=7 you can also specify a register as a variable. For example SETTILE (R1,R2)=8: for the tile that will be changed the X-coordinate is in Register #1 and de Y-coordinate is in Register #2

Comments

Use ; to add comments.

Example:

```
CLEARSCREEN()   ; this clears the screen
```

Todo

These are the main features that will be attended next (in no particular order):

- Tiles that can shoot bullets or beams (static, like a turret)
- Place tiles on sprite layer with transparency
- Add moving enemies which can shoot
- Add a scoring mechanic
- Add hud-features
- Add subroutine options in scripting language
- Perform tile animations in two directions
- Set amount of tiles that should be used for player walking animations (currently fixed at two)
- Support more image sizes in SHOWIMG
- Make 'modding' possible, in a way that anyone can write assembly to enhance the engine with additional functionality.
- Enhance the build in script editor
- Implement 'volume' on playsound for zsm files