Vircon32:ゲームの作り方

資料作成日2021.05.19年Carra著

これは何だ?

このドキュメントは、Vircon32仮想コンソール用のゲームの作成を開始するためのクイックガイドです。目標は、ゼロから開始し、一連の手順を実行して、非常に基本的なゲームの作成方法を知ることです。

まとめ

このガイドは小さなセクションで構成されており、最初に全体的なレベルでのプロセスを説明し、後でゲームに必要な個々の側面(画像、サウンド、プレイヤーのコントロールなど)に焦点を当てます。

[Introduction 2](#_Toc72326827)

[Development tools 2](#_Toc72326828)

[Packing a ROM 4](#_Toc72326829)

[Automating the process 5](#_Toc72326830)

[Typical game structure 6](#_Toc72326831)

[Interpreting compiler errors 8](#_Toc72326832)

[Reading the state of gamepads 9](#_Toc72326833)

[Drawing characters and objects 9](#_Toc72326834)

[Scrolling backgrounds 11](#_Toc72326835)

[Sound effects 11](#_Toc72326836)

[Background music 12](#_Toc72326837)

[Handling game objects 12](#_Toc72326838)

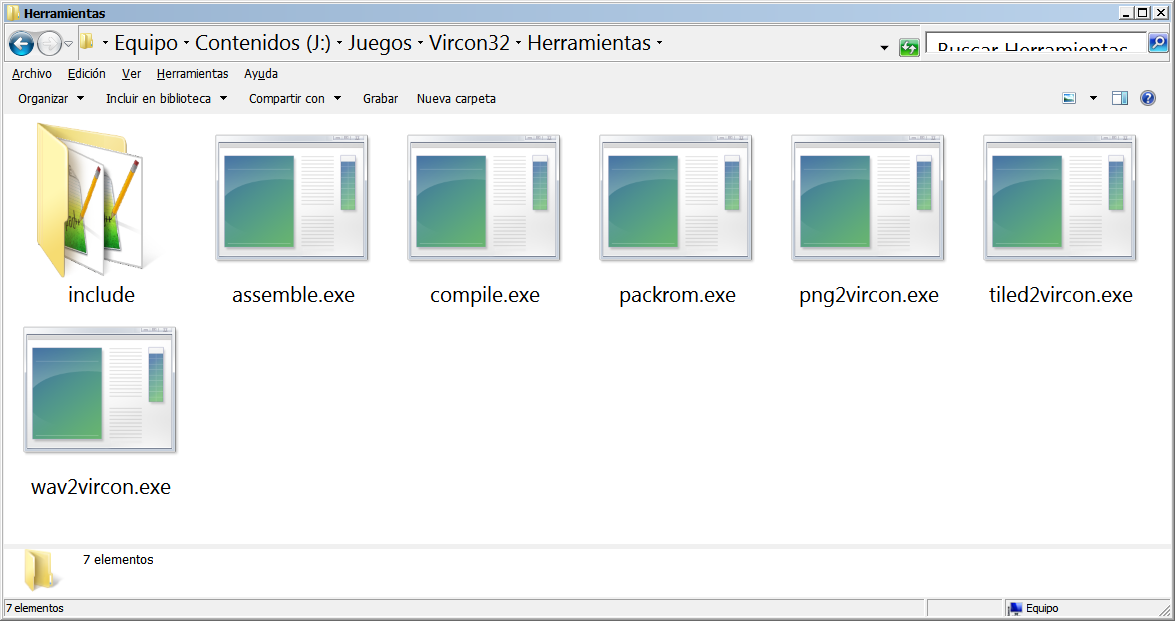
[Where to keep learning 13](#_Toc72326839)

## はじめに

このガイドでは、Cプログラムを記述し、開発ツールを使用してコンソール上で実行する実際のプロセスに重点を置いています。また、いくつかのコンソール機能の使用方法の典型的な例も示しますが、このガイドでは、Vircon32で実行できるすべてのことについて詳細には触れません。コンソールシステムをよりよく理解するために、このガイドの動作方法を読むことをお勧めします。

## 開発ツール

Vircon32の開発ツールをダウンロードすると、次の要素を含むフォルダが表示されます。



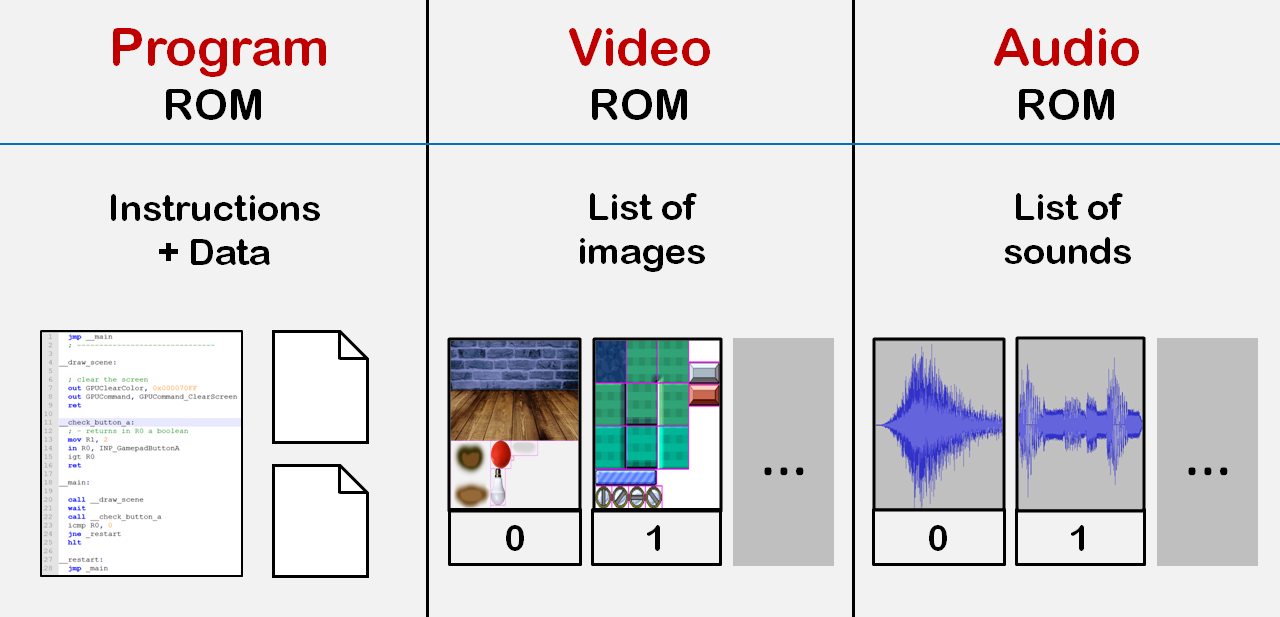
これらの実行可能プログラムはすべてコマンドラインプログラムであり、スクリプトを記述するか、開発環境(IDE)などの他のプログラムからこれらのツールを使用することによって自動化できます。これを行うには、通常、PATH変数を使用して、Windows実行可能パスにDevToolsフォルダを追加する必要があります。

これらの各要素を見てみましょう。

* **コンパイル.exe**はCコンパイラです。C言語プログラムをアセンブリコードに変換します。
* **assemble.exe(英語の可能性**はアセンブラです。これはアセンブリコードをVircon32 CPUが理解できるバイナリ命令に変換します。
* **packrom.exe**は、ROM Packerです。バイナリプログラムを取得し、使用する必要のあるイメージとサウンドを追加します。その結果、Virconエミュレータがロードできる単一の.v32ファイルが作成されます。
* **png2vircon.exe(英語の可能性あり)**は、PNG形式のイメージを、Vircon32グラフィックスチップが使用できる内部形式に変換します。
* **wav2vircon.exe<ファイル名>**は、WAV形式のサウンドをVircon32サウンドチップで使用できる内部形式に変換します。
* **tiled2vircon.exe<ファイル名>**は、タイルエディタで作成されたタイルマップをインポートするツールです。これは、各レイヤをバイナリファイルに変換します。このファイルをカートリッジに含めて、2D配列として使用できます。
* **次のものが含まれ**このフォルダには、コンパイラとプログラムが使用する標準Cライブラリのファイルが含まれています。

### プログラムを作成するステップ

Vircon32コンソールは、他のエミュレータと同様にROMファイルである仮想カートリッジで動作します。各ゲームは、ゲームに必要なすべてのものがパックされた単一のファイルです。このコンテンツは、次の3つの部分に分かれています。



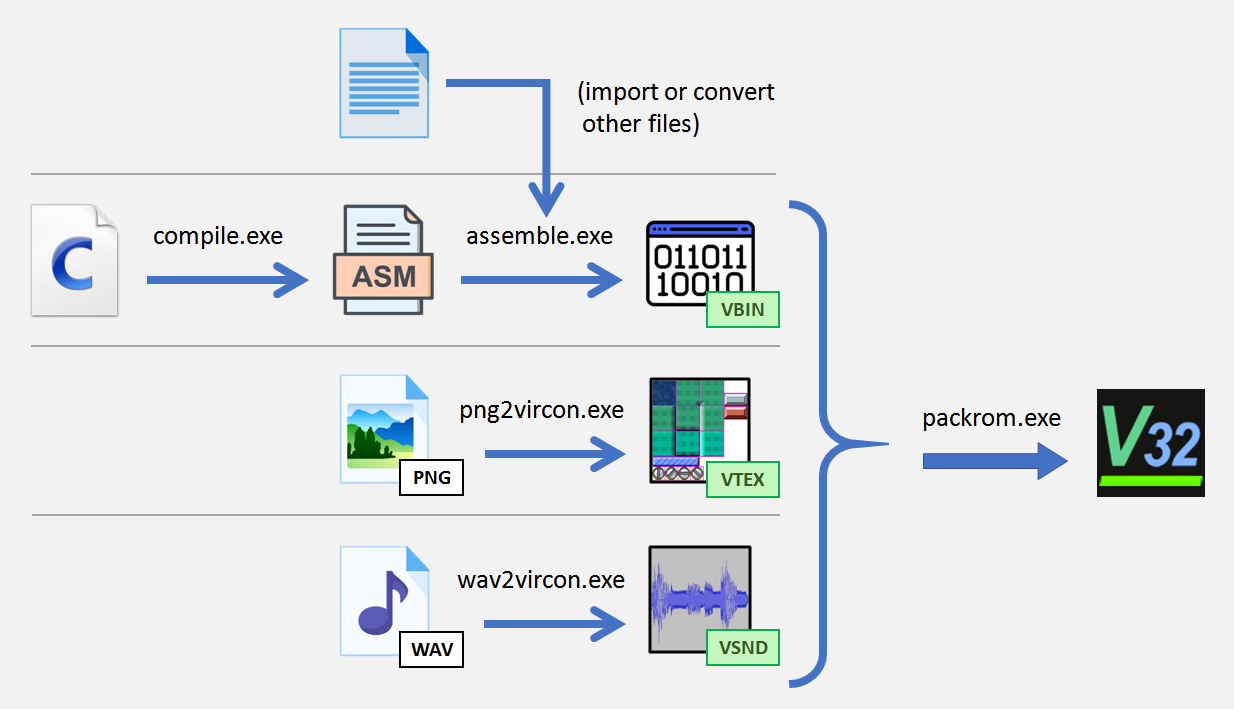
Vircon32カートリッジにプログラムのみが含まれている場合、コードからVircon32ゲームを作成するには、次の2つの手順を実行するだけです。



しかし、Vircon32では、これを使ってプログラムromを作成しただけです。また、プログラムで使用できるように、一連のイメージとサウンドをカートリッジに含める必要があります。一般的に、次の手順に従います。

1. Cプログラムのコンパイルとアセンブル
2. すべてのイメージをVircon32のネイティブ形式に変換
3. すべてのサウンドをVircon32のネイティブ形式に変換する
4. Packerを使用して、すべてを最終的なROMファイルに結合します。
5. エミュレータを使用してプログラムをテストする

また、インポートする必要のある追加のデータファイル(タイルマップなど)がある場合もあります。一般に、プロセスは次のモデルに従います。



緑でマークされたファイルは、すでにネイティブのVircon32フォーマットであるため、最終的なパッキングに含めることができます。

## ROMのパッキング

packerは、ROMに多くのファイルを含める必要がある場合があります(Vircon32ゲームは、最大256の画像と1024のサウンドを持つことができます)。そのため、コマンドラインを介してファイルパスを提供することは、あまり実用的ではありません。

代わりに、ROMの完全な定義を含むXMLファイルを入力パラメータとして使用します。これには、タイトルやバージョンなどの一般的な詳細も含まれます。例では、ROMがどのように使用されているかを確認できますが、XMLドキュメントの形式は次のようになります。

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" standalone="no" ?>  **<ROM定義>**バージョン="1.0"を**>**  **<メモリ**タイプ="カートリッジ"タイトル="ゲームタイトル"バージョン="1.0"を**/>(英語の可能性**  **<バイナリ**パス="PathToBinary.vbin"**/>(英語の可能性**  **<テクスチャ>**  **<テクスチャ**パス="PathToTexture1.vtex"**/>(英語の可能性**  **<テクスチャ**パス="PathToTexture2.vtex"**/>(英語の可能性**  **</textures><テクスチャ>**  **<サウンド>**  **<サウンド**パス=『PathToSound1.vsnd』**/>(英語の可能性**  **<サウンド**パス=『PathToSound2.vsnd』**/>(英語の可能性**  **</サウンド>**  </rom-definition><パス> |

テクスチャとサウンドが表示される順序は、ファイルにパックされる順序と同じです。つまり、このプログラムは、リスト内のID 0の最初のテクスチャ、ID 1の2番目のテクスチャなどにアクセスできます。

プログラムにはテクスチャもサウンドもありません。プログラムバイナリは、必須の3つの部分のうちの唯一のものです。ただし、XMLには常に<テクスチャ>および<サウンド>エレメント(空の場合も含む)

## プロセスの自動化

Packerは、プログラムがすでにコンパイルされ、サウンドとイメージがすでに変換された状態で、ゲームを作成するための最後のステップを実行するだけです。ゲームを作成するということは、変更を行うたびにROMを再作成することを意味するため、通常はスクリプトまたは外部ツールを使用して、1回のクリックですべてを自動化する必要があります。

最も簡単な方法は、手順を実行するための小さなコマンドラインスクリプトを作成することです。実用的な例は付属のプログラムソースにありますが、ここではWindows BATコマンドファイルのこの簡単な例を示します。

|  |
| --- |
| @echoオフ  エコーCコードをコンパイルする  エコー--------------------------  **コンパイル**Program.c-o obj\Program.asm goto:が失敗しました  エコーASMコードのアセンブル  エコー--------------------------  **アセンブル**obj\Program.asm-o obj\Program.vbin goto:失敗しました  エコーPNGテクスチャを変換する  エコー--------------------------  **png2vircon(英語の可能性あり)**Texture.png-o obj\Texture.vtex goto:失敗  echo WAVサウンドを変換する  エコー--------------------------  **wav2vircon(wav 2ビデオ会議)**Sound.wav-o obj\Sound.vsnd goto:失敗  エコーROMをパックする  エコー--------------------------  **packrom**RomDefinition.xml-o bin\FinalGame.v32 goto:失敗しました  goto:成功しました  :失敗  echo構築に失敗しました  終了/b%errorlevel%  :成功しました  echo構築に成功しました  /bを終了  @echoオン |

このスクリプトは単純化することができますが、重要なのは、すべてのステージを呼び出し、いずれかのステージでエラーが発生した場合には、プロセスを続行しようとするのではなく停止することです。

このスクリプトでは、サブディレクトリobjとbinを使用していることもわかります。Cでのプログラミング(Vircon32だけでなく)では、通常、すべてのプロジェクトに対してこのようなフォルダ構造を使用します。このようにして、最終的な結果(「bin」内)、生成される中間ファイル(「obj」内)、およびプロジェクトのソースファイル自体を分離できます。

## 一般的なゲーム構造

Cプログラムからゲームを作成する方法はすでにわかっています。次に、プログラム自体の作成方法について説明します。これは、プログラミングを行ったことのない人にプログラミングを教えることを目的としたものではなく、ゲームのプログラミング方法に関する完全なガイドでもありません。ただし、例を使用していくつかの基本的な概念を示し、ゲームをコーディングするときに通常使用されるプラクティスをいくつか確認します。

一般に、Vircon32ゲームは通常、次の構造に従って記述されます。

* ライブラリを含める
* テクスチャ、領域、サウンドに名前を付ける
* グローバル変数の宣言
* 補助機能
* 主な機能
* テクスチャとサウンドを設定する
* ゲームを初期化する
* メインループ
* ゲームパッドを読み込み、プレイヤーのアクションを適用する
* ゲームの仕組みをシミュレートする
* シーンとシーン内のオブジェクトを画面上に描画する
* 次のフレームまで待機してゲームの速度を制御する

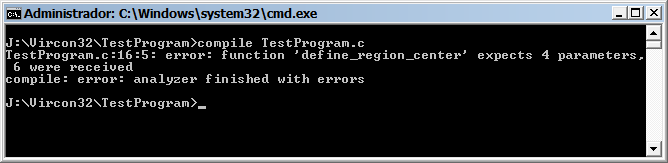
この構造の最も重要な要素は、メインループです。通常、ゲームは常に無限ループで繰り返され、このループ内のアクションはフレームごとに1回実行されます。Vircon32では、これは1秒間に60回繰り返すことを意味します。

次のページでは、このプログラム構造に続く小さな完全な例を見ることができます。このプログラムは非常に単純ですが(固定された背景に文字を描画するだけで、それを移動できます)、実際には1つのページで構造を見ることができます。さらに、さまざまな部分にコメントが付けられています。

|  |
| --- |
| //Virconライブラリを含める  #include"ビデオ.h"  #include"入力.h"  #include"時間.h"  //--------------------------------------------------------------------------------------------------  //定義  //テクスチャ領域に名前を付ける  #define RegionBackground 0  #RegionRobotRight 1を定義  //--------------------------------------------------------------------------------------------------  //メイン関数  空所メイン(空所)を  {を  //--------------------------------------------------------------------------------------------------  //パート1:テクスチャの設定    select\_texture(0);選択テクスチャ    //テクスチャ領域を定義する  select\_region(RegionBackground);  define\_region\_topleft(0,0,639,359);定義領域の左上  select\_region(RegionRobot);  define\_region(1,361,66,441,33,417);定義領域    //--------------------------------------------------------------------------------------------------  //パート2:初期化    //私たちのロボットはスクリーンの中心から  内部RobotX=スクリーン幅/2;  内部RobotY=スクリーンの高さ/2;    //--------------------------------------------------------------------------------------------------  //パート3:メインループ    //フレーム(60 fps)ごとにゲームロジックを繰り返します。  一方(の真)を  {を  //押した方向にロボットを移動  内部方向X、方向Y;  gamepad\_direction(&DirectionX,&DirectionY);<方向>  RobotX+=2\*方向X;  RobotY+=2\*方向Y;    //画面いっぱいに背景を描画する  select\_region(RegionBackground);  draw\_region\_at(0,0);と入力します。    //現在の位置にロボットを描画します。  select\_region(RobotImage);  draw\_region\_at(RobotX,RobotY);と入力します。    //ゲームの速度を制御するために次のフレームを待機  終了フレーム();  }をクリックし  }をクリックし |

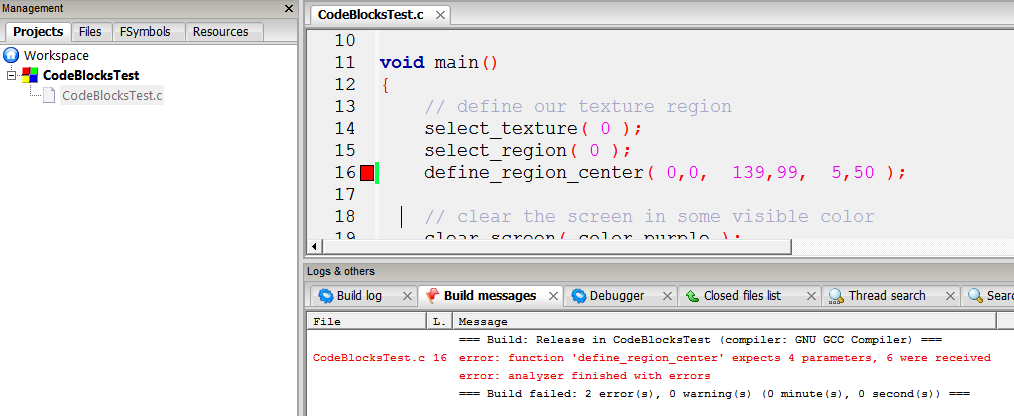
## コンパイラエラーの解釈

私たちがプログラムでミスをするのは普通のことなので、コンパイラが私たちに警告したときにエラーを見つける方法を知ることに興味があります。Vircon32 Cコンパイラはgccコンパイラと同じ形式でエラーを出力します。つまり、ファイル名、行、列です。次の例でその様子を見ることができます。



IDEなしでプログラムを作成する場合は、Notepad++などの行番号を含むテキストエディタを使用して、これらのエラーを手動で検索できます。ただし、コンパイラは(ある程度)gccのように動作するように設計されているため、IDEを正しく設定することでIDEを操作できます。

ダウンロード可能なサンプルソースの中に、「CodeBlocksTest」という名前のものがあります。これは、Code::Blocks用に作成されたプロジェクトです。これを使用すると、プログラムをコンパイルできるだけでなく、[実行]ボタンを使用してエミュレータでROMをテストすることもできます。このイメージでは、Code::Blocksが同じエラーをどのように表示するかを確認できます。



このメソッドは他のIDEではテストされていませんが、他のIDEでも動作する可能性は十分にあります。このサンプルプロジェクトでは、Cファイルを設定して、Code::Blocksが**しない**代わりに、コンパイル後のイベントとしてBATファイル(前述のようなもの)を実行するようIDEに指示します。

これにより、グローバルレベルでプログラムを操作する方法がすでにわかりました。次のセクションでは、ゲームで通常必要とされるいくつかのアクションについて説明し、それらを実行する方法を示します。

## ゲームパッドの状態を読み取る

通常、メインループで最初に行う必要があることは、ゲームを更新するためにプレイヤーがゲームパッドで何を押しているかを知ることです。ここでは、2つの選択肢を示します:個々の方向をチェックするか、それらをグローバルにチェックします。

|  |
| --- |
| //ここでは、垂直方向を無視してX方向にのみ移動します。  もし(gamepad\_left()>0)位置x-=速度;  もし(gamepad\_right()>0)位置x+=速度;  //ここでは、自動的に任意の方向に移動します。  gamepad\_direction(&DirectionX,&DirectionY);<方向>  Position.x+=速度\*方向X;  Position.y+=速度\*方向Y; |

ボタンはどうなりますか?Vircon32にはイベントがありません。ボタンがすでに押されたのか、今押されたのかを知るには、通常、前のフレームの状態を保存して、それが変更されたかどうかを確認する必要があります。ただし、これは必要ではありません。コンソールにはすでにこの情報が表示されています。次のようにして検出できます。

|  |
| --- |
| //ボタンAの状態を読み取る  内部ボタンA=gamepad\_button\_a();  //Aを押すと、エネルギーの充電が開始されます。  もし(ボタンA==1)  StoredEnergy=1;  //フレームごとに押されたままになり、エネルギーが増加する  else if文(ButtonA>0)。  StoredEnergy++;  //ボタンを離したときに撮影  もし(ボタンA==-1)  シュート(); |

## 文字やオブジェクトを描画する

ゲームでは、キャラクターがアニメーションを持つことはよくあります。そのための簡単な方法は、いくつかの連続した領域を定義し、経過フレームに従ってそれらのすべてを循環させることである。

たとえば、次のように描かれたキャラクタがあるとします。



Cライブラリを使用すると、これらの連続する領域を一度に定義できます。この例では、これがどのように行われるか、また、描画時にどのようにアニメートされるかを確認できます。

|  |
| --- |
| //アニメーションの領域を定義する  定義領域マトリックス  (の  100//作成する最初の領域のID  1,1//最初の領域の左上隅  50,50人//最初の領域の右下コーナー  25,47人//最初の領域の参照点  7,1//マトリックスにはXに7つの領域、Yに1つの領域があります。  1個//領域は1ピクセルの分離ボーダーを持つ  )  //アニメーションイメージを5フレームごとに変更  AnimationTime++;  もし(AnimationTime%5==0)  アニメーションイメージ++;  //アニメーションが終了すると、最初から繰り返されます。  もし(アニメーションイメージ>=7)  アニメーションイメージ=0;  //アニメーションを描画する各フレーム  select\_region(100+AnimationImage);選択範囲  draw\_region\_at(PlayerX、PlayerY); |

多くのゲームでは、キャラクターは両側を向いて描かれるのではなく、常に右を向いて描かれます。次に、それらを左を向くようにするために、イメージが反転されます。これは、Xのスケーリングが-1に等しい場合に実現できます。

|  |
| --- |
| //右向きの文字を描く  もし(PlayerSpeedX>=0)  draw\_region\_at(PlayerX、PlayerY);  //左向きの文字を描く  それ以外  {を  set\_drawing\_scale(-1,1);図面尺度設定  draw\_region\_zoomed\_at(PlayerX、PlayerY);  }をクリックし |

## 背景のスクロール

Vircon32で使用されているテクスチャ(1024x1024ピクセル)は画面(640x360)よりも大きいですが、それでも大きなシナリオを作成するには小さすぎます。コンソールでは、複数のテクスチャを使用してそれらを接続できますが、多くのゲームで最もよく使用される方法は、タイルマップを作成することです。

DevToolsに含まれるプログラムには、次のものがあります。tiled2vircon.exe<ファイル名>このプログラムを使用すると、エディタTiledで作成されたタイルマップをインポートできます。これを行うには、Cプログラムをコンパイルする前にインポータツールを呼び出し、コンパイル時にインポートされたファイルがすでに作成されるようにします。インポートツールを使用して、プロジェクトのobjフォルダに「Level1.map」というファイルを作成したとします。次に、Cから、これを次のような配列であるかのようにプログラムにインポートできます。

|  |
| --- |
| //ファイルの内容が配列としてカートリッジに保存される  埋め込みint【TilesInY】【TilesInX】マップブリック="obj\\Level1.map(オブジェクト\\レベル1.マップ)" |

次に、テクスチャには、マップを描画するための各タイルのイメージが含まれます。プログラムでタイルのセットを定義するには、関数を使用します。関数define\_region\_matrix()前と同じです。

その後、画面にマップを描画するために、XとYのマップをウォークスルーするループを使用できます。ただし、注意が必要です。マップは非常に大きくなる可能性があり、すべてのフレームを描画しようとすると、コンソールからの要求が多すぎる可能性があります。その場合は、画面に表示されるタイルの範囲を確認し、ループをXとYのその範囲に制限する必要があります。

## 効果音

Vircon32には、サウンドを再生するための16のチャネルがあります。自動的に再生するオプションがあります(この機能は空きチャネルを検索します)。

|  |
| --- |
| //任意の空きチャンネルでサウンドを再生します。  play\_sound(SoundExplosion); |

もう1つのオプションは、いくつかのサウンドチャネルを手動で予約することです。たとえば、キャラクターが話すためのチャネルが1つしかない場合は、2つのフレーズが同時に再生されないようにします(これは奇妙に聞こえます)。

|  |
| --- |
| //前の文字サウンドを停止する  もし(get\_channel\_state(ChannelPlayer)!=チャネル停止)  stop\_channel(ChannelPlayer);  //新しいサウンドを再生できるようになりました。  play\_sound\_in\_channel(SoundHello、ChannelPlayer); |

## BGM

通常、私たちは音楽のために特定の既知のチャネルを予約することに関心があります。このようにして、私たちは必要に応じて音楽を一時停止または再開できます。また、音楽が連続的に再生され、ループで繰り返されることも一般的です。ここでは、両方のことを示します:

|  |
| --- |
| //この音楽をループで再生するように設定する  select\_sound(MusicLevel1);<サウンド>  set\_sound\_loop(真);  //ループを有効にして音楽を再生します。  play\_sound\_in\_channel(サウンドをチャンネルで再生)MusicLevel1ChannelMusic)  //音楽チャネルを停止する  stop\_channel(ChannelMusic); |

## ゲームオブジェクトの操作

より複雑なゲームでは、プレーヤー、ショット、さまざまなタイプの敵など、フレームごとに更新および描画する必要がある多種多様なオブジェクトが存在する可能性があります。C++では、これらのゲームエンティティをより一般的な方法で処理するためにクラス継承を使用するのが一般的です。Cでは、ユニオンを使用してそれを実現できます。この小さな例では、変数データを使用してオブジェクトを定義する方法を示します。

|  |
| --- |
| //異なるデータを持つ構造を定義する  //各オブジェクトが必要とするものに応じて  構造体EnemyState{.};(敵の状態)  構造体BulletState{.};  //この共用体は、任意のオブジェクトのデータを格納できます。  ユニオンオブジェクトの状態  {を  EnemyState AsEnemy  BulletState AsBullet  }  //実際に何が含まれているかを知るために、ユニオンに情報を追加します  構造体ゲームオブジェクト  {を  ブールアクティブ  内部オブジェクトタイプ;  オブジェクトデータの状態;  }をクリックし |

次に、そのデータの処理方法を確認する必要があります。現在、Vircon32コンパイラはまだ制限されており、ダイナミックメモリ(リストを使用できないことを意味する)や関数へのポインタもサポートされていない。ただし、次に示すように、配列を使用して、より一般的なレベルでオブジェクトを処理することができる。

|  |
| --- |
| //存在可能なオブジェクトの最大数を持つ配列を使用します。  GameObject[50]すべてのオブジェクト;  //次のようにオブジェクトを処理します。  対象(の内部i=0;i<50;i++)  {を  ゲームオブジェクト\*ThisObject=ゲームオブジェクト[i];    もし(!ゲームオブジェクト[i]->アクティブ)  続ける    //オブジェクトのタイプを選択する  もし(オブジェクトタイプ==TypeEnemy)  ゲームオブジェクト[i]->アクティブ=ProcessEnemy(ゲームオブジェクト[i]->AsEnemy);    それ以外もし(オブジェクトタイプ==TypeBullet)  ゲームオブジェクト[i]->アクティブ=ProcessBullet(ゲームオブジェクト[i]->AsBullet);  }をクリックし |

## どこで学習を続けるか

これまで、ゲームを作成するために必要な基本について説明してきましたが、さらに多くのことを学ぶ必要があります。ここで説明したことを実践するには、チュートリアルからプログラムをコンパイルして実験することから始めます。

後で、前進して新しいアイデアを考えるための良い方法は、Vircon32のための他のゲームやプログラムがどのように作られているかを見ることです。利用可能なテストプログラムのソースコードは、より高度な例として役立ちます。なぜなら、それらはまた、非常に多くのコメントを有し、通常、ここに示されるのと同じ構造に従うからです。

コンソールの異なるシステムがどのように動作するかを詳細に知ることも、将来のためにお勧めします。これは、この同じ文書セットの対応するガイドで説明されています。