En este tutorial crearemos el clásico jueguito Tetris utilizando el compilador WxDev C++, que puedes descargar en este enlace.

El artículo y código original pertenecen a Jan Bodnar, quien hace tutoriales sobre programación.

Para este videojuego, se utilizan algunas figuras denominadas como Tetrominoes, que tienen forma de S, de Z, de T, de L, lineal, L invertida y una forma cuadrada. Estas figuras bajan en la pantalla del juego y el objetivo es acomodarlas sin que queden espacios vacíos en cada línea.

WxWidgets es un toolkit que nos sirve para crear aplicaciones con interfaz gráfica, aunque existen algunas librerías para crear videojuegos más avanzados, con este toolkit podemos crear aplicaciones sencillas, incluyendo este juego.

Para las figuras no se utilizan imágenes, en lugar de eso utilizamos la API de dibujo que incluye WxWidgets Programming Tool. Para que el juego se ejecute como un ciclo, se utiliza wxTimer.

Código fuente

Archivo Shape.h

```
#ifndef SHAPE_H
#define SHAPE_H

enum Tetrominoes { NoShape, ZShape, SShape, LineShape,
TShape, SquareShape, LShape, MirroredLShape };

class Shape
{
```

```
public:
Shape() { SetShape(NoShape); }
void SetShape(Tetrominoes shape);
void SetRandomShape();

Tetrominoes GetShape() const { return pieceShape; }
int x(int index) const { return coords[index][0]; }
int y(int index) const { return coords[index][1]; }

int MinX() const;
int MaxX() const;
int MinY() const;
int MaxY() const;

Shape RotateLeft() const;
Shape RotateRight() const;

private:
void SetX(int index, int x) { coords[index][0] = x; }
void SetY(int index, int y) { coords[index][1] = y; }
Tetrominoes pieceShape;
int coords[4][2];
};

#endif
```

Archivo Shape.cpp

```
#include <stdlib.h>
#include <algorithm>
#include «Shape.h»
```

```
using namespace std;

void Shape::SetShape(Tetrominoes shape)
{
static const int coordsTable[8][4][2] = {
{ { 0, 0 }, { 0, 0 }, { 0, 0 }, { 0, 0 } },
{ { 0, -1 }, { 0, 0 }, { -1, 0 }, { -1, 1 } },
{ { 0, -1 }, { 0, 0 }, { 1, 0 }, { 1, 1 } },
{ { 0, -1 }, { 0, 0 }, { 0, 1 }, { 0, 2 } },
{ { -1, 0 }, { 0, 0 }, { 1, 0 }, { 0, 1 } },
{ { 0, 0 }, { 1, 0 }, { 0, 1 }, { 1, 1 } },
{ { -1, -1 }, { 0, -1 }, { 0, 0 }, { 0, 1 } },
{ { 1, -1 }, { 0, -1 }, { 0, 0 }, { 0, 1 } }
};

for (int i = 0; i < 4 ; i++) {
for (int j = 0; j < 2; ++j)
coords[i][j] = coordsTable[shape][i][j];
}
pieceShape = shape;
}

void Shape::SetRandomShape()
{
int x = rand() % 7 + 1;
SetShape(Tetrominoes(x));
}

int Shape::MinX() const
{
int m = coords[0][0];
for (int i=0; i<4; i++) {
```

```
m = min(m, coords[i][0]);
}
return m;
}

int Shape::MaxX() const
{
int m = coords[0][0];
for (int i=0; i<4; i++) {
m = max(m, coords[i][0]);
}
return m;
}

int Shape::MinY() const
{
int m = coords[0][1];
for (int i=0; i<4; i++) {
m = min(m, coords[i][1]);
}
return m;
}

int Shape::MaxY() const
{
int m = coords[0][1];
for (int i=0; i<4; i++) {
m = max(m, coords[i][1]);
}
return m;
}
```

```
Shape Shape::RotateLeft() const
{
if (pieceShape == SquareShape)
return *this;

Shape result;
result.pieceShape = pieceShape;
for (int i = 0; i < 4; ++i) {
result.SetX(i, y(i));
result.SetY(i, -x(i));
}
return result;
}

Shape Shape::RotateRight() const
{
if (pieceShape == SquareShape)
return *this;

Shape result;
result.pieceShape = pieceShape;
for (int i = 0; i < 4; ++i) {
result.SetX(i, -y(i));
result.SetY(i, x(i));
}
return result;
}
```

Estos códigos corresponden a las figuras del juego, sus tamaños y colores.

Archivo Board.h

```
#ifndef BOARD_H
#define BOARD_H

#include "Shape.h"
#include <wx/wx.h>

class Board : public wxPanel
{

public:
Board(wxFrame *parent);
void Start();
void Pause();
void linesRemovedChanged(int numLines);

protected:
void OnPaint(wxPaintEvent& event);
void OnKeyDown(wxKeyEvent& event);
void OnTimer(wxCommandEvent& event);

private:
enum { BoardWidth = 10, BoardHeight = 22 };

Tetrominoes & ShapeAt(int x, int y) { return board[(y *
BoardWidth) + x]; }
```

```
int SquareWidth() { return GetClientSize().GetWidth() /
BoardWidth; }
int SquareHeight() { return GetClientSize().GetHeight() /
BoardHeight; }
void ClearBoard();
void DropDown();
void OneLineDown();
void PieceDropped();
void RemoveFullLines();
void NewPiece();
bool TryMove(const Shape& newPiece, int newX, int newY);
void DrawSquare(wxPaintDC &dc, int x, int y, Tetrominoes
shape);

wxTimer *timer;
bool isStarted;
bool isPaused;
bool isFallingFinished;
Shape curPiece;
int curX;
int curY;
int numLinesRemoved;
Tetrominoes board[BoardWidth * BoardHeight];
wxStatusBar *m_stsbar;
};

#endif
```

Archivo Board.cpp

```
#include "Board.h"

Board::Board(wxFrame *parent)
: wxPanel(parent, wxID_ANY, wxDefaultPosition,
wxDefaultSize, wxBORDER_NONE)
{
timer = new wxTimer(this, 1);

m_stsbar = parent->GetStatusBar();
isFallingFinished = false;
isStarted = false;
isPaused = false;
numLinesRemoved = 0;
curX = 0;
curY = 0;

ClearBoard();

Connect(wxEVT_PAINT, wxPaintEventHandler(Board::OnPaint));
Connect(wxEVT_KEY_DOWN, wxKeyEventHandler(Board::OnKeyDown));
Connect(wxEVT_TIMER, wxCommandEventHandler(Board::OnTimer));
}


void Board::Start()
{
```

```
if (isPaused)
return;

isStarted = true;
isFallingFinished = false;
numLinesRemoved = 0;
ClearBoard();

NewPiece();
timer->Start(300);
}

void Board::Pause()
{
if (!isStarted)
return;

isPaused = !isPaused;
if (isPaused) {
timer->Stop();
m_stsbar->SetStatusText(wxT("paused"));
} else {
timer->Start(300);
wxString str;
str.Printf(wxT("%d"), numLinesRemoved);
m_stsbar->SetStatusText(str);
}
Refresh();
}
```

```
void Board::OnPaint(wxPaintEvent& event)
{
wxPaintDC dc(this);

wxSize size = GetClientSize();
int boardTop = size.GetHeight() - BoardHeight *
SquareHeight();


for (int i = 0; i < BoardHeight; ++i) {
for (int j = 0; j < BoardWidth; ++j) {
Tetrominoes shape = ShapeAt(j, BoardHeight - i - 1);
if (shape != NoShape)
DrawSquare(dc, 0 + j * SquareWidth(),
boardTop + i * SquareHeight(), shape);
}
}

if (curPiece.GetShape() != NoShape) {
for (int i = 0; i < 4; ++i) {
int x = curX + curPiece.x(i);
int y = curY - curPiece.y(i);
DrawSquare(dc, 0 + x * SquareWidth(),
boardTop + (BoardHeight - y - 1) * SquareHeight(),
curPiece.GetShape());
}
}
}

void Board::OnKeyDown(wxKeyEvent& event)
```

```
{
if (!isStarted || curPiece.GetShape() == NoShape) {
event.Skip();
return;
}

int keycode = event.GetKeyCode();

if (keycode == 'p' || keycode == 'P') {
Pause();
return;
}

if (isPaused)
return;

switch (keycode) {
case WXK_LEFT:
TryMove(curPiece, curX - 1, curY);
break;
case WXK_RIGHT:
TryMove(curPiece, curX + 1, curY);
break;
case WXK_DOWN:
TryMove(curPiece.RotateRight(), curX, curY);
break;
case WXK_UP:
TryMove(curPiece.RotateLeft(), curX, curY);
break;
case WXK_SPACE:
```

```
DropDown();
break;
case 'd':
OneLineDown();
break;
case 'D':
OneLineDown();
break;
default:
event.Skip();
}

}

void Board::OnTimer(wxCommandEvent& event)
{
if (isFallingFinished) {
isFallingFinished = false;
NewPiece();
} else {
OneLineDown();
}
}

void Board::ClearBoard()
{
for (int i = 0; i < BoardHeight * BoardWidth; ++i)
board[i] = NoShape;
}
```

```
void Board::DropDown()
{
int newY = curY;
while (newY > 0) {
if (!TryMove(curPiece, curX, newY - 1))
break;
--newY;
}
PieceDropped();
}

void Board::OneLineDown()
{
if (!TryMove(curPiece, curX, curY - 1))
PieceDropped();
}

void Board::PieceDropped()
{
for (int i = 0; i < 4; ++i) {
int x = curX + curPiece.x(i);
int y = curY - curPiece.y(i);
ShapeAt(x, y) = curPiece.GetShape();
}

RemoveFullLines();

if (!isFallingFinished)
NewPiece();
}
```

```
void Board::RemoveFullLines()
{
int numFullLines = 0;

for (int i = BoardHeight - 1; i >= 0; --i) {
bool lineIsFull = true;

for (int j = 0; j < BoardWidth; ++j) {
if (ShapeAt(j, i) == NoShape) {
lineIsFull = false;
break;
}
}

if (lineIsFull) {
++numFullLines;
for (int k = i; k < BoardHeight - 1; ++k) {
for (int j = 0; j < BoardWidth; ++j)
ShapeAt(j, k) = ShapeAt(j, k + 1);
}
}
}

if (numFullLines > 0) {
numLinesRemoved += numFullLines;
wxString str;
str.Printf(wxT("%d"), numLinesRemoved);
m_stsbar->SetStatusText(str);

isFallingFinished = true;
```

```
curPiece.SetShape(NoShape);
Refresh();
}
}

void Board::NewPiece()
{
curPiece.SetRandomShape();
curX = BoardWidth / 2 + 1;
curY = BoardHeight - 1 + curPiece.MinY();

if (!TryMove(curPiece, curX, curY)) {
curPiece.SetShape(NoShape);
timer->Stop();
isStarted = false;
m_stsbar->SetStatusText(wxT("game over"));
}
}

bool Board::TryMove(const Shape& newPiece, int newX, int newY)
{
for (int i = 0; i < 4; ++i) {
int x = newX + newPiece.x(i);
int y = newY - newPiece.y(i);
if (x < 0 || x >= BoardWidth || y < 0 || y >= BoardHeight)
return false;
if (ShapeAt(x, y) != NoShape)
return false;
}
```

```
curPiece = newPiece;
curX = newX;
curY = newY;
Refresh();
return true;
}

void Board::DrawSquare(wxPaintDC& dc, int x, int y,
Tetrominoes shape)
{
static wxColour colors[] = { wxColour(0, 0, 0), wxColour(204,
102, 102),
wxColour(102, 204, 102), wxColour(102, 102, 204),
wxColour(204, 204, 102), wxColour(204, 102, 204),
wxColour(102, 204, 204), wxColour(218, 170, 0) };

static wxColour light[] = { wxColour(0, 0, 0), wxColour(248,
159, 171),
wxColour(121, 252, 121), wxColour(121, 121, 252),
wxColour(252, 252, 121), wxColour(252, 121, 252),
wxColour(121, 252, 252), wxColour(252, 198, 0) };

static wxColour dark[] = { wxColour(0, 0, 0), wxColour(128,
59, 59),
wxColour(59, 128, 59), wxColour(59, 59, 128),
wxColour(128, 128, 59), wxColour(128, 59, 128),
wxColour(59, 128, 128), wxColour(128, 98, 0) };


wxPen pen(light[int(shape)]);
```

```
pen.SetCap(wxCAP_PROJECTING);
dc.SetPen(pen);

dc.DrawLine(x, y + SquareHeight() - 1, x, y);
dc.DrawLine(x, y, x + SquareWidth() - 1, y);

wxPen darkpen(dark[int(shape)]);
darkpen.SetCap(wxCAP_PROJECTING);
dc.SetPen(darkpen);

dc.DrawLine(x + 1, y + SquareHeight() - 1,
x + SquareWidth() - 1, y + SquareHeight() - 1);
dc.DrawLine(x + SquareWidth() - 1,
y + SquareHeight() - 1, x + SquareWidth() - 1, y + 1);

dc.SetPen(*wxTRANSPARENT_PEN);
dc.SetBrush(wxBrush(colors[int(shape)]));
dc.DrawRectangle(x + 1, y + 1, SquareWidth() - 2,
SquareHeight() - 2);
}
```

Estos códigos corresponden a los controles de juego, donde se especifica qué teclas sirven para realizar los movimientos.

Archivo Tetris.h

```
#include <wx/wx.h>
```

```
class Tetris : public wxFrame
{
public:
Tetris(const wxString& title);

};
```

Archivo Tetris.cpp

```
#include "Tetris.h"
#include "Board.h"

Tetris::Tetris(const wxString& title)
: wxFrame(NULL, wxID_ANY, title, wxDefaultPosition,
wxSize(220, 380))
{
wxStatusBar *sb = CreateStatusBar();
sb->SetStatusText(wxT("0"));
Board *board = new Board(this);
board->SetFocus();
board->Start();
}
```

Archivo main.h

```
#include <wx/wx.h>

class MyApp : public wxApp
{
public:
virtual bool OnInit();

};
```

Archivo main.cpp

```
#include «main.h»
#include «Tetris.h»

IMPLEMENT_APP(MyApp)

bool MyApp::OnInit()
{
srand(time(NULL));
Tetris *tetris = new Tetris(wxT(«Tetris»));
tetris->Centre();
tetris->Show(true);

return true;
}
```

Al compilar y ejecutar el programa, el juego se inicia automáticamente. Para pausar, se presiona la tecla *p*, con la *barra espaciadora*, la pieza en juego baja rápidamente al fondo, con la letra *d*, la pieza baja más rápido.

El array de coordenadas da la forma a la pieza del juego, por ejemplo, el arreglo {0,-1}, {-1,1}, {0,1}, {0,0} representa la figura cuadrada.

En el compilador, debemos crear todos los archivos con los códigos de arriba y añadirlos al proyecto.

Luego, damos clic en compilar y ejecutar

Entonces ya podremos disfrutar del juego. Para más detalles sobre el código, visita el artículo original.

Si requieres algún programa en específico, no dudes en contactarnos aquí para una cotización.