Tetris Project

Carlos VALADARES et Yaya KAMISSOKHO

April 9, 2019

1 Interface Utilisateur

1.1 Aire d'informations

Cette partie de l'interface nous permet d'afficher les informations utiles du jeu, à savoir le **score**, la vitesse de descente des tetraminos représenté noté **level** et aussi le nombre des **lignes** déjà casseés.Comme montré par la figure 1.

1.2 Aire du Jeu

Dans cette partie de l'interface on peut visualiser le jeu, c'est à travers cette region qu'on observe la grille du jeu et les tetraminos ansi que leurs deplacement.

1.3 Aire de la caméra

La dernière region de l'interface concerne la caméra. Dans cette partie on peut voir l'image utilisé par la detection des gester pour translater et rotacioner les tetraminos, dans cette region on peut observer aussi la detection des gestes en temps d'execution simbolisé par une carré verte dans la region detecté.

2 Conception des classes

Nous avons créer des classes le plus independant possible, ainsi chaque classe créé a son role bien defini.

2.1 TetrisMainWindow

Ce classe concerne la fenêtre principalle, elle est responsable pour être un conteiner pour toutes les régions et widgets de l'interface décrites précedement. En autre, c'est elle que fait l'instanciation des classes GameControl et VideoControl bien que gérer les touches du clavier et emettre les signais respectives à la classe GameControl, au cas où l'utilisateur joue à travers du clavier:

- A: Déplacement du tetraminos à gauche
- D: Déplacement du tetraminos à droite
- R: Rotation du tetraminos au sense horaire
- F: Augmenter la vitesse(le niveau)

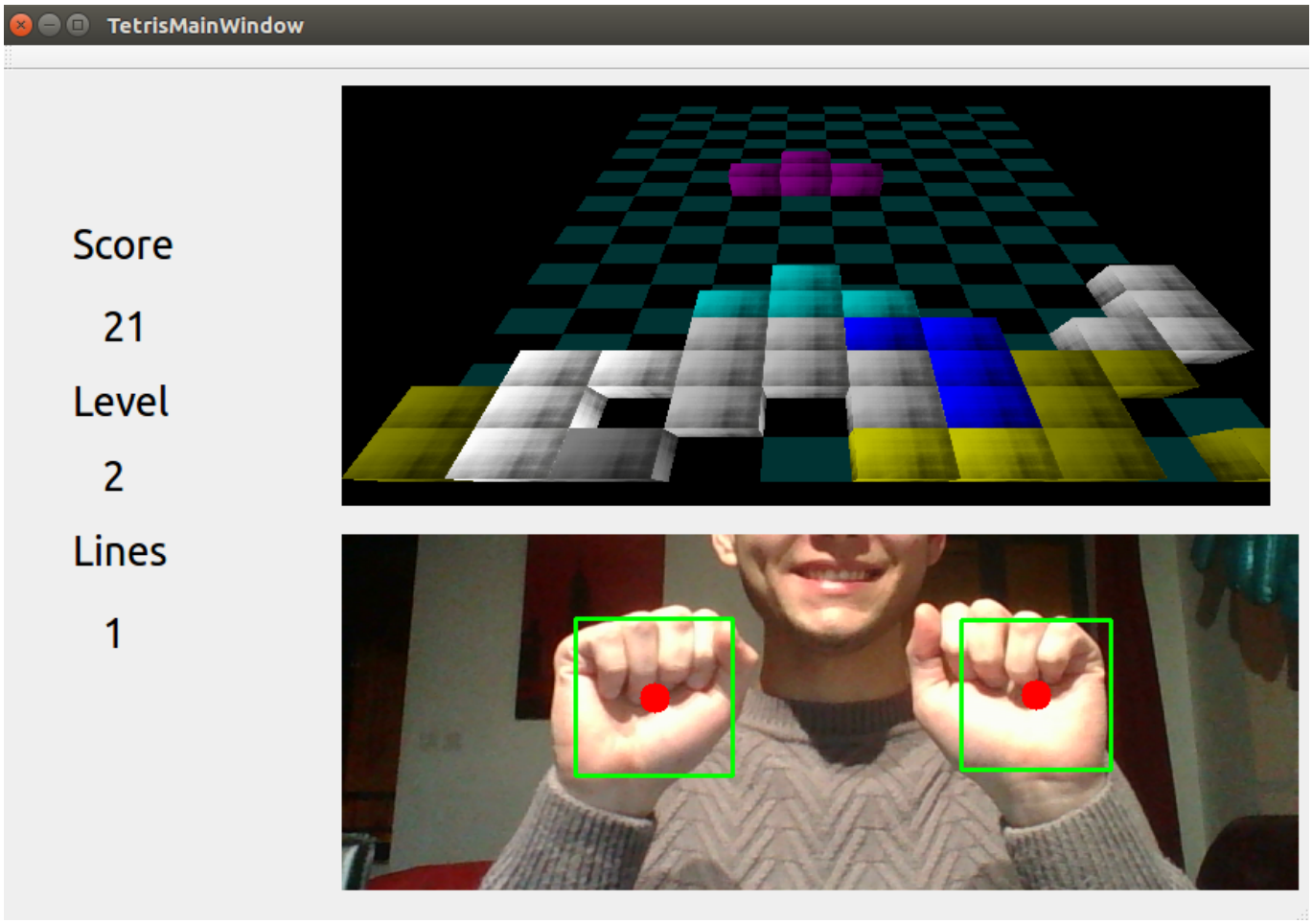

Figure 1: Image de l'interfqce utilisateur

2.2 GameControl

La classe GameControl est responsable de la gestion du jeu, c'est-à-dire, la creation des tetraminos, la creation de la grille du jeu, augmentation de la vitesse du jeu, faire translater et rotationer les tetraminos en accord avec le signal reçu(soit par la classe TetrisMainWindow si on joue sur le clavier soit par la classe VideoControl si on joue à travers des gestes), detection des lignes remplies et les cassés.

2.3 VideoControl

La classe VideoControl est responsable de la partie Video, c'est-à-dire, la capture des images, la detection des gestes et aussi, l'affichage de l'image dans sa respective région dans l'interface.

2.4 PaintWidget

La classe PaintWidget gére toute l'afichage correspondant au jeu. Ainsi, cette classe, herité de QOpenGLWidget, affiche toutes les tetraminos et la grille en se refraichant souvent pour aficher correctement toutes les etats du jeu.

2.5 Tetramino

La classe Tetramino est responsable d'encapsuler toutes les informations intrinsèques à chaque tetramino comme sa couleur, ses quatre cellules ocupés dans le tableu du

jeu, ses rotations, les indices de sa position dans la grille(ligne et colonne) bien que sa translation par rapport au repère de la grille 3D(Point en haut à gauche de la grille).

2.6 Cellule

La classe cellule répresent chaque case(cellule) du tableau du jeu. Donc, elle contien quatre points 3D qui répresent les quatre points du carré, la ligne et la colonne ocupé dans la grille pour éviter la perte d'information une fois que une ligne est supprimé. En plus, elle stock aussi un status pour dire si une cellule est occupé par un tetramino, de cette façon nous pouvons facilement détecter des lignes remplies.

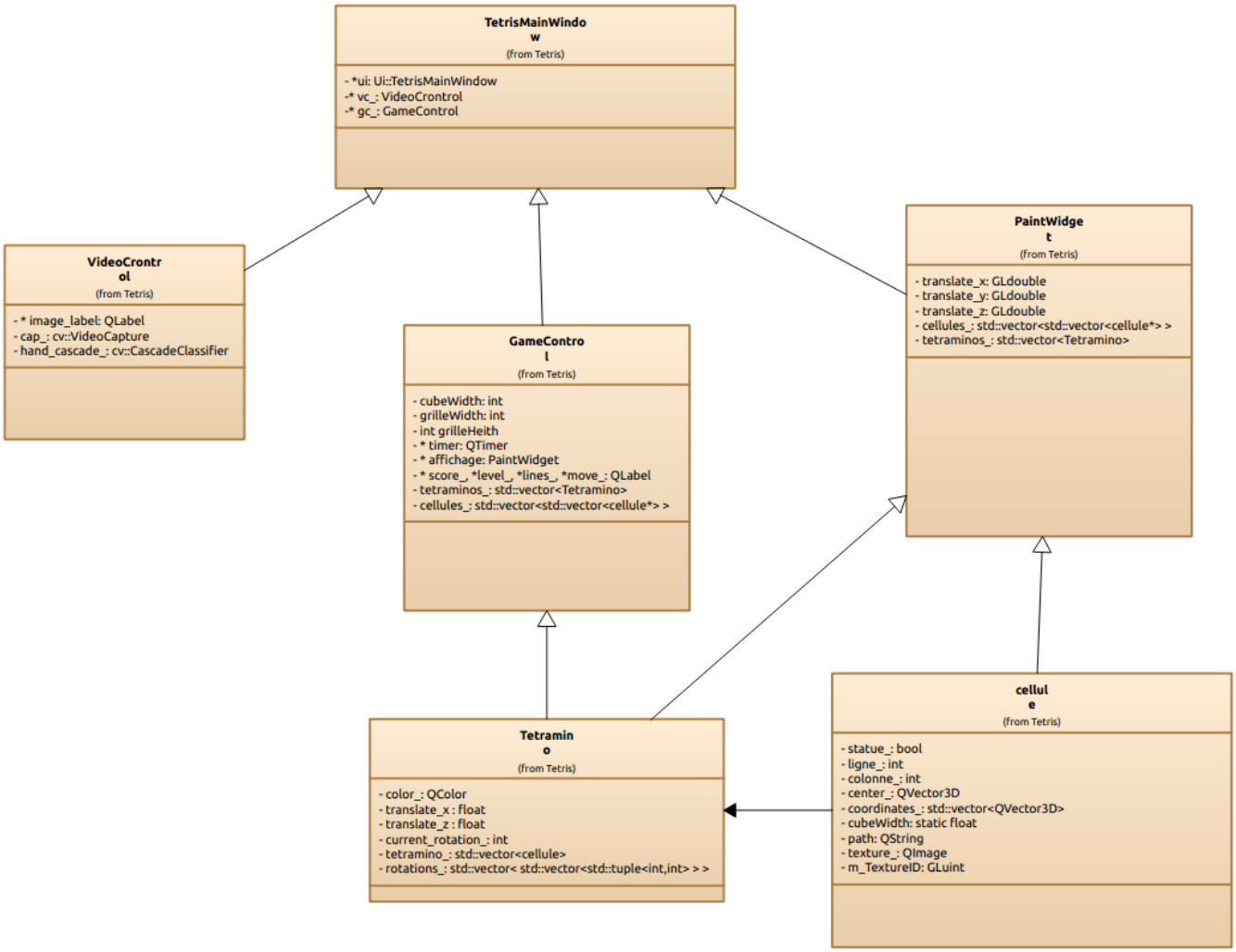

Figure 2: Diagramme de classe

diagrame de classe

3 Etat de finalisation de l'application

Au Moment de la livrasion du projet les features que sont testés et validés sont:

- Affichage de la grille 3D incliné.
- Creation d'un tetraminos et son affichage.

- Arrete d'un tetraminos lorsqu'il arrive em bas du tableau ou il n'est peut plus bouger.
- Detection des mains et gestes produites par l'utilisateur.
- Lieson entre le jeu et les gestes detectés.
- Bouger et rotacioner le tetramino en fonction des gestes ou touches de clavier respectives.
- Detection de la fin du jeu et son reinitialization
- Augmentation de difilculté du jeu par l'augmentation de la vitesse
- Counting le score et les lignes casséés.
- Ajoute des textures aux tetraminos

Cependant il y a des features que lors de la livraison du projet que n'est marchait pas ou que générait des bogues et par consequence elles sont enlèves du projet:

- Augmentation de la vitesse en clicant sur un RadioButton dans l'interface
- Affichage du prochain tetraminos

4 Fichiers d'entete des classes

Les fichiers d'entete avec des commentaires eplicatives de toutes les classes peuvent etre consultés ci dessus

4.1 TetrisMainWindow

```
#ifndef TETRISMAINWINDOW_H
#define TETRISMAINWINDOW_H

#include <QMainWindow>
#include<QWidget>
#include "videocrontrol.h"
#include "gamecontrol.h"


namespace Ui {
class TetrisMainWindow;
}

class TetrisMainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit TetrisMainWindow(QWidget *parent = nullptr);
    ~TetrisMainWindow();

private:
    Ui::TetrisMainWindow *ui;
    VideoCrontrol* vc_;             //variable pour instancier la classe VideoControl
    GameControl* gc_;               //variable pour instancier la classe GameControl

protected:
    void keyPressEvent(QKeyEvent* event);
```

```
29
30 signals:
31     void LeftSignal();              //Signal pour notifier le GameControl pour faire
       la translation vers la gauche du tetramino courant
32     void RightSignal();             //Signal pour notifier le GameControl pour faire
       la translation vers la droite du tetramino courant
33     void RotateSignal();            //Signal pour notifier le GameControl pour
       rotationer le tetramino courant
34     void PauseSignal();
35     void MoveSlowOrSpeed();         //Signal pour notifier le GameControl pour
       modifier la vitesse
36     void changeNiveau(int);
37
38 };
39
40 #endif // TETRISMAINWINDOW_H
```

4.2 GameControl

```
1  #ifndef GAMECONTROL_H
2  #define GAMECONTROL_H
3
4  #include <QObject>
5  #include "paintwidget.h"
6  #include <QWidget>
7  #include <QLabel>
8  #include <QRadioButton>
9
10 class GameControl : public QObject
11 {
12     Q_OBJECT
13 public:
14     //constructeur
15     explicit GameControl(PaintWidget* p = nullptr, QLabel* score = nullptr, QLabel*
       level = nullptr,
16                          QLabel* lines = nullptr, QObject *parent = nullptr);
17
18 signals:
19
20 public slots:
21     void incrementZ();       //slot appell  pour translater le tetramino dans le
       tableau
22     void LeftRequest();      //Requete pour bouger le tetramino a gouche
23     void RightRequest();     //Requete pour bouger le tetramino a droite
24     void RotateRequest();    //Requete pour rotacioner le tetramino
25     void Pause();            //Requete pour faire une pause dans le jeu
26     void MoveSlowOrSpeed(); //Requete pour changer la vitesse du jeu
27
28 private:
29     bool c;
30    int cubeWidth;
              //Largeur de chaque case de la grille
31    int grilleWidth;
              //Largeur de la grille
32    int grilleHeith;
              //Hauteur de la grille
33    int pauseTime_;
              //variable pour stocker la vitesse du jeu
34    int maxLine_;
              //Ligne o  se situe le derni re tetraminos(plus en haut) pour otimiser la
       recherche des lignes remplies
```

```
35      std::vector<int> remplissage_;
36      QTimer* timer;
                //Variable pour appeller les foncions    chaque intervale
37      PaintWidget* affichage;
                //Variable responsable de l'affichage
38      QLabel* score_, *level_, *lines_, *move_;
                //Labels de l'interface    mettre    jour pour la classe GameControl
39      std::vector<Tetramino> tetraminos_;
                //Vecteur de tetraminos existants
40      std::vector<std::vector<cellule*> > cellules_;
                //Tableu du jeu
41
42      std::vector< std::vector<std::vector<std::tuple<int, int> > > > AllTetraminos_;
                //Ensemble de toutes les tetraminos avec tous ses rotations possibles
43      void createTetramino();
                //Fonction pour cr er un tetraminos
44      void createAllTetraminos();
                //Fonction pour cr er le tableu que stock toutes les tetraminos possibles
           instancier
45      void createGrille();
                //Fonction pour cr er la grille du jeu
46
47      /*
48        fonction qui permet de voir si un tetramino peut bouguer ou pas
49      */
50      bool canWeMoveDown();
51      bool canWeMoveLeft();
52      bool canWeMoveRight();
53      bool canWeRotate();
54
55      /*
56        detecter la fin de la patie
57      */
58      bool endGame();
59
60      /*
61        recommencer la partie
62      */
63      void RestartGame();
64
65      /*
66         Cass  une ligne du tableau qui est remplie
67       */
68      void RemoveLine(int i);
69      std::vector<QString> listeImage;
70  };
71
72  #endif // GAMECONTROL_H
```

4.3 VideoControl

```
1   #ifndef VIDEOCRONTROL_H
2   #define VIDEOCRONTROL_H
3   #include<QWidget>
4   #include <QLabel>
5   #include <QMainWindow>
6   #include "opencv2/imgproc/imgproc.hpp"
7   #include "opencv2/highgui/highgui.hpp"
8   #include <opencv2/objdetect/objdetect.hpp>
9   #include <opencv2/core/core.hpp>
10  #include "opencv2/opencv.hpp"
```

```
#include <QTimer>
#include <QDebug>

class VideoCrontrol : public QObject
{
    Q_OBJECT
public:
    explicit VideoCrontrol(QLabel* i, QObject *parent = nullptr);

private:
    int count_;
    int timing_response_;                          //le temps qu'on envoie un signal
    QLabel* image_label;                           //Label de l'interface pour afficher l'image
    cv::VideoCapture cap_;                         //Variable pour faire des capture des images    travers de la cam ra
    cv::CascadeClassifier hand_cascade_;           //Variable pour stocker le classifieur
signals:
    void VideoLeftSignal();                        //Fonction qui notifie la classe GameControl qu'elle a detect  un geste pour translater le tetramino vers la gauche
    void VideoRightSignal();                       //Fonction qui notifie la classe GameControl qu'elle a detect  un geste pour translater le tetramino vers la driote
    void VideoRotateSignal();                      //Fonction qui notifie la classe GameControl qu'elle a detect  un geste pour rotationer le tetramino

public slots:
    void cameraThread();                           //Fonction appel  dans chaque 100ms pour s'occuper de capturer des images et detecter des gestes
};

#endif // VIDEOCRONTROL_H
```

4.4 paintWidget

```
#ifndef PAINTWIDGET_H
#define PAINTWIDGET_H

#include <QWidget>
#include <QGLWidget>
#include <QMouseEvent>
#include <QTimer>
#include <QVector>
#include <GL/glu.h>
#include <QColor>
#include <QOpenGLWidget>
#include<qopenglfunctions.h>
#include"cellule.h"
#include<iostream>
#include"tetramino.h"
#include<QVector3D>
#include<QDebug>

class PaintWidget : public QOpenGLWidget
{
    Q_OBJECT
public:
    explicit PaintWidget(QWidget* parent = nullptr);

```

```
    /*
        Fonctions pour partager des variables entre les classes
    */
    void SetTetraminosVector(std::vector<Tetramino> &v);
    void setGrille(std::vector<std::vector<cellule*> > & grille);
    void setParametersGrille(int gW, int gH, int ml);
protected:
    // Fonction d'initialisation
    void initializeGL();

    // Fonction de redimensionnement
    void resizeGL(int width, int height);

    // Fonction d'affichage
    void paintGL();

    // Fonction de gestion d'interactions clavier
    void keyPressEvent(QKeyEvent * event);

public slots:
    void moveCircle(int);
private:
    int grilleWidth_;
    int grilleHeith_;
    int maxLine_;
    QTransform transform_;
    int centre_x = 150;
    int centre_y = 150;
    GLdouble translate_x=0;
    GLdouble translate_y=0;
    GLdouble translate_z=0;
   int width = 5; // default width of 5
    int    length = 5; // default length of 5
     int   height = 12; // default height of 12
    std::vector<std::vector<cellule*> > cellules_;
    std::vector<Tetramino> tetraminos_;
    Tetramino *t;
    double cubeWidth=10;

    /*
      fonction qui dessine la grille
    */
    void drawEnvironment();

    /*
      fonction qui dessine les tetraminos
    */
    void drawTetraminos();

    /*
      fonction qui creer les tetraminos
    */
    void creatTetramino();
};

#endif // PAINTWIDGET_H
```

4.5 Tetramino

```
#ifndef TETRAMINO_H
#define TETRAMINO_H
```

```
#include "cellule.h"
#include <QColor>
#include <iostream>
#include<QDebug>

class Tetramino
{
public:
    Tetramino(std::vector<cellule> positions, std::vector< std::vector<std::tuple<int,int>>> rot, QColor c);
    void draw();                    //fonction qui dessine le tetramino
    void rollTetramino();           //fonction qui translate le tetramino vers le bas
    void moveLeft();                //fonction qui translate le tetramino vers la gauche
    void moveRight();               //fonction qui translate le tetramino ver la droite
    void Rotate();                  //fonction qui rotatione le tetramino
    //draw
private:
    QColor color_;                  //couleur du tetramino
    float translate_x ;             //sa translation de l'axe X par rapport au r pere de la grille
    float translate_y;
    float translate_z ;
    int current_rotation_;                                                  //rotation courant
    std::vector<cellule> tetramino_;                                        //son vector de 4 cellules
    std::vector< std::vector<std::tuple<int,int>>> rotations_;              //toutes les rotations possibles du tetramino

public:

    /*
      fonctions qui de translation
    */
    float getTranslateZ();
    float getTranslateX();
    float getTranslateY();
    void translateZ();
    void translateX(QString sens);
    void translateY();


    std::vector<std::tuple<int,int>> afterRotation();       //retourne les coordonn s apr s une rotation hipotetique pour utiliser dans la fonction canWeRotate()
    std::vector<cellule> getCellules();
};

#endif // TETRAMINO_H
```

4.6 Cellule

```
#ifndef CELLULE_H
#define CELLULE_H

#include<QColor>
#include<qdebug.h>
#include<QVector3D>
#include<iostream>
```

```
8  #include <QWidget>
9  #include <QGLWidget>
10 #include <QMouseEvent>
11 #include <QTimer>
12 #include <QVector>
13 #include <GL/glu.h>
14 #include <QColor>
15 #include <QOpenGLWidget>
16 #include<qopenglfunctions.h>
17 #include<iostream>
18 class cellule
19 {
20 public:
21
22     //---------------- Constructeur ------------------
23
24     cellule(QVector3D p1,QVector3D p2,QVector3D p3,QVector3D p4,int ligne,int colonne
       );
25
26 private:
27
28     //---------------- Variables qui pour representer une case de la grille
       ----------------
29     bool statue_;
30     QColor color_;
31     int ligne_;
32     int colonne_;
33     QVector3D center_;
34     std::vector<QVector3D> coordinates_;
35     float minX;
36     float minZ;
37     float planY;
38
39     //---------------- la taille de nos cub (case du grille) ------------------
40     static float cubeWidth;
41
42     //---------------- parametre qui represante la texture ------------------
43     QString path;
44     QImage texture_;
45     GLuint m_TextureID ;
46
47 public:
48
49     //---------------- definition des fonctions ------------------
50     /*
51       permet d'envoyer la statue(un boolean) de la grille (true si la case et occuper
       sinon false)
52     */
53     bool getStatue(){return  statue_;}
54
55     /*
56       permet de changer la statue de la case
57     */
58     void setStatue(bool b){statue_=b;}
59
60     /*
61       definire la color de la cellule
62     */
63     void setColor(QColor c){color_=c;}
64
65     /*
```

```
66        returne la couleur, ligne colonne, coordonner
67      */
68      QColor getColor(){return color_;}
69      int getLigne(){return ligne_;}
70      int getColonne(){return colonne_;}
71       std::vector<QVector3D> getCoordinates(){return  coordinates_;}
72
73      /*
74        recuperer le centre d'un cellule
75      */
76      QVector3D getCenter(){return center_;}
77
78      /*
79        incrementer ou decremanter la ligne ou la colonne
80      */
81      void rollLigne(){ligne_+=-1;}
82      void moveLeft(){colonne_ -= 1;}
83      void moveRight(){colonne_ += 1;}
84      void rotate(float Xpivot, float Ypivot);
85
86      /*
87        dessiner une case
88      */
89      void draw();
90
91  };
92
93
94  #endif // CELLULE_H
```

Lien pour le projet sur github: `https://github.com/cfcv/Tetris`