MMS PROJEKT
3D3S

Renato Babojelić, Vitomir Čanadi, Zvonimir Iveković
Mentor: Goran Igaly

18. svibnja 2016.

1 UVOD

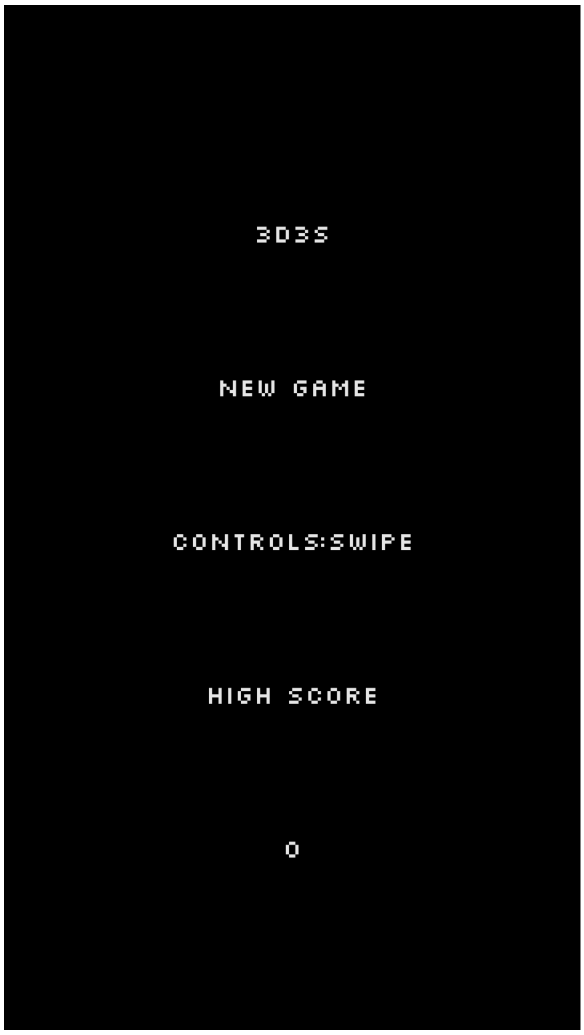
Kao završni projekt na kolegiju Multimedijski sustavi, razvili smo igru 3D3s. Riječ je o klonu trodimenzionalne varijante popularne igre Tetris. Mehanka igre prirodno se proširuje na trodimeznìnalni prostor; blokovi `L,I` i `S` iz klasične varijante igre 'dobili' su još jednu dimenziju i sada su izgrađeni od četiri jedinične kocke, dok je pravokutni blok postao kocka izgrađena od osam jediničnih. Blokovi simetrični `L` i `S` oblicima, koje znamo iz originala, ovdje su samo simetrične slike osnovnih,do kojih je moguće doći rotacijama, pa ih nije potrebno posebno definirati. Neke varijate trodimenzionalnog tetrisa za blokove uzimaju sve tetrakocke[1]; što je svakako u duhu originala, no što, smatramo, previše otežava igru. Nadalje, blokove koje slažemo padaju po jednoj osi, dok ih je moguće pomicati i smještati s obzirom na druge dvije osi. Rotacija blokova, za razliku od klasičnog Tetrisa, ovdje je moguća oko sve tri osi, dok se odsjecanje nivoa događa kada je cijela horizontalna ravnina popunjena jediničnim kockama.

Igra je izrađena u Unity[2] *game engine*-u i prilagođena za Android mobilne uređaje, na kojima je i detaljno testirana. Iako izrađena s Android sustavom na umu, 3D3s je moguće izgraditi za bilo koju drugu platformu uz minimalne modifikacije koda koji upravlja korisnikovim unosom (Npr. promjena touchscreen unosa na unos preko tipkovnice i sl.). Upravo zbog portabilnosti projekata na mnoge platforme, kao i zbog brojne i stručne zajednice korisnika, odabrali smo Unity za izradu ovog projekta.

[1]Tijela sastavljena od točno četiri jednake kocke, međusobno spojene zajedničkim stranama.
[2]Unity je višeplatformski *engine* za razvoj video igara. Podržava izradu igara za PC, mobilne uređaje i aktualne igraće konzole.

2 KAKO IGRATI

Pokretanjem igre otvara se glavni izbornik kao na slici 1. Klikom na 'New game' započinjemo igru, dok klikom na 'Controls' mijenjamo način pomicanja blokova, bilo potezima prsta po ekranu u 'Swipe' modu, ili preko gumbiju na ekranu u 'Buttons' modu. Broj ispod 'High score' predstavlja, do sada, najveći broj sakupljenih bodova u jednoj igri.

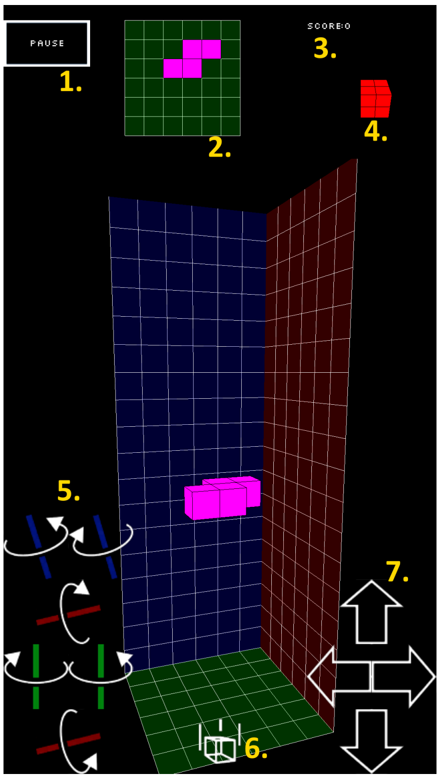


Slika 1: Glavni izbornik

Nakon klika na 'New game', ako smo odabrali 'Buttons' kontrole, otvara se ekran kao na slici 2. Elementi na ekranu, redom kako su enumerirani, su:

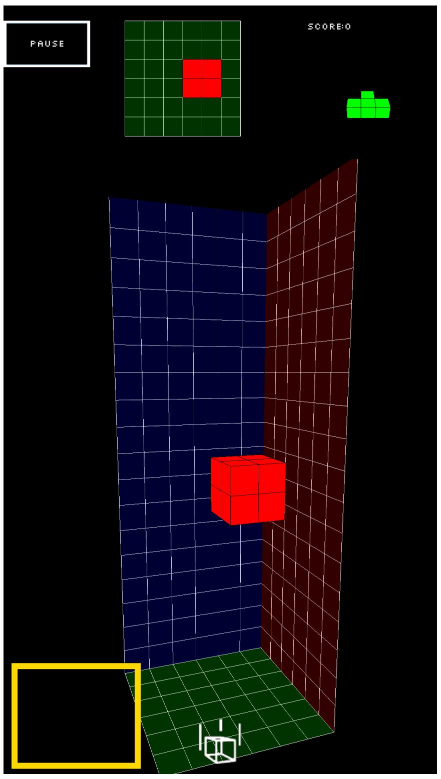
1. gumb 'Pause' pauzira igru,
2. pogled odozgo, projekcija svih blokova na horizontalnu(zelenu) ravninu,
3. broj bodova sakupljenih unutar trenutne igre,

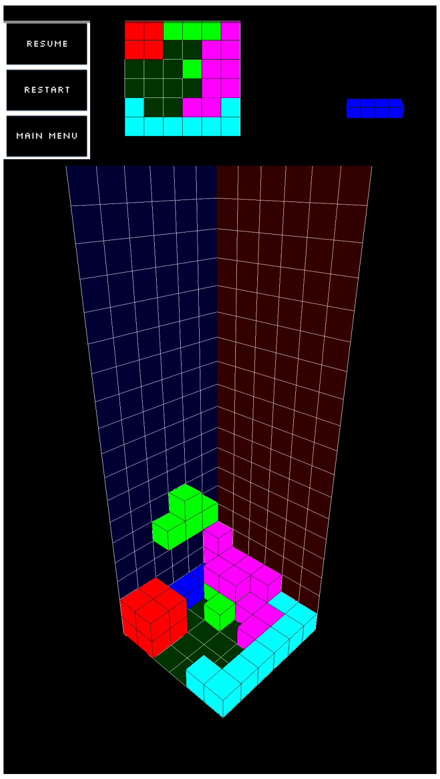
4. blok koji je sljedeći na redu,
5. kontrole za upravljanje rotacijom, boja označava rotaciju bloka oko normale ravnine iste boje,
6. ubrzava pad trenutnog bloka,
7. strelice za pomicanje.



Slika 2: 'Button' kontrole

Ako su pak odabrane kontrole 'Swipe', na ekranu se ne iscrtavaju strelice za pomicanje ili gumbi za rotaciju. Blokovi se pomiću potezima gore, dolje, lijevo, desno, a rotiraju istim potezima sa istovremeno dodirnutim donjim djelom ekrana(žuti pravokutnik), kao na slici 3.

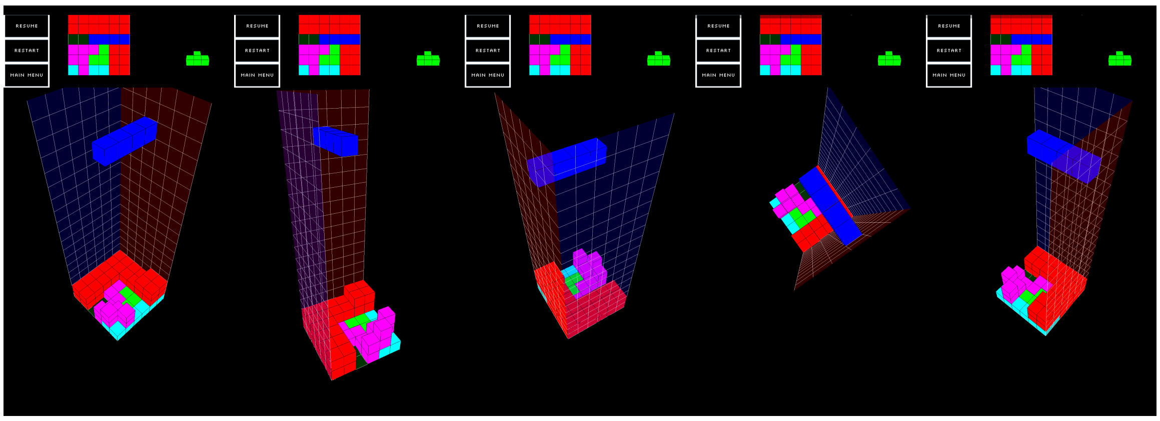




Slika 3: 'Swipe' kontrole

Slika 4: Pauzirana igra

Pauziranjem igre, neovisno o kontrolama, prikazuje se ekran, slika 4. Dodirom na gumbe s vrha ekrana; 'Resume', 'Restart', 'Main menu', redom nastavljamo igru, započinjemo novu ili se vraćamo na glavni izbornik. Dok je igra pauzirana potezima ljevo-desno, odnosno gore-dolje rotiramo scenu oko vertikalne, odnosno horizontalne osi(slika 5).



Slika 5: Rotiranje scene

3 IMPLEMENTACIJA

3.1 class Block

Kako smo spomenuli u uvodu, igraci blokovi izgrađeni su od jednakih 'jediničnih' kocaka. Svaka kocka jednoznačno je određena položajem jednog svog vrha, pa je stoga predstavljena jednim objektom klase `Vector3int`, tj. trodimenzionalnim vektorom s cjelobrojnim koeficjentima. Blokovi su sada skupovi susjednih vektora, npr. blok `L` definiran je ovako.

Ova klasa implementira osnovne blokove, njihove relativne koordinate, pomicanje i rotaciju.

```
new Block( Vector3int.Zero,1,
        new Vector3int(0,0,0), new Vector3int(1,0,0),
        new Vector3int(-1,0,0),new Vector3int(1,0,1))
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;
using UnityEngine;


namespace Tetris {
 class Block {
  Vector3int position;
  public List < Vector3int > bricks = new List < Vector3int >
     ();
  int color = 0;


  public Block(Vector3int position, int color,
   params Vector3int[] bricks) {
   this.position = position;
   this.bricks = bricks.ToList < Vector3int > ();
   this.color = color;
  }
  private Block(Block block) {
   this.position = block.position;
   foreach(Vector3int brick in block.bricks)
   this.bricks.Add(new Vector3int(brick));
   this.color = block.color;
  }
  private Block(Vector3int position, Block block) {
```

```
  this.position = position;
  foreach(Vector3int brick in block.bricks)
  this.bricks.Add(new Vector3int(brick));
  this.color = block.color;
 }

 // ima smisla samo za brickse
 public Vector3int
 AbsPos(Vector3int brick) {
  return this.position + brick;
 }

 //korisiti se kod Bounded(..) tako da izracuna relativne
    kordinate granica(lower,upper)
 //umjesto da racuna puno apsolutnih koordinata od bricksova
 public Vector3int
 RelPos(Vector3int v) {
  return v - this.position;
 }

 //local rotation
 public void Rotate(Vector3int a) {
  bricks.ForEach(v => v.Rotate(a));
 }
 public Block GetRotated(Vector3int a) {
  Block ret = new Block(this);
  ret.Rotate(a);
  return ret;
 }

 public void Place(Vector3int position) {
  this.position = position;
 }
 public void Move(Vector3int dPos) {
  this.position += dPos;
 }

 public void MoveRight() {
  this.Move(Vector3int.Right);
 }
 public void MoveLeft() {
  this.Move(Vector3int.Left);
 }
 public void MoveUp() {
```

```
 this.Move(Vector3int.Up);
}
public void MoveDown() {
 this.Move(Vector3int.Down);
}
public void MoveForward() {
 this.Move(Vector3int.Forward);
}
public void MoveBackward() {
 this.Move(Vector3int.Backward);
}

//neda mi se ostale
public Block GetMovedDown() {
 Block ret = new Block(this);
 ret.MoveDown();
 return ret;
}
public Block GetMoved(Vector3int dir) {
 Block ret = new Block(this);
 ret.Move(dir);
 return ret;
}


//unutar kvadra definiranog vrhovima (0,0,0) i upper
public bool
BoundedZeroAndUpper(Vector3int upper) {
 return bricks.All(v => v.BoundedHalf(
  this.RelPos(Vector3int.Zero), this.RelPos(upper)));
}
public bool
NotBoundedZeroAndUpper(Vector3int upper) {
 return BoundedZeroAndUpper(upper) == false;
}

public static bool
CollisionTest(Block b1, Block b2) {
 return b1.bricks.Any(v =>
  b2.bricks.Any(u => b1.AbsPos(v) == b2.AbsPos(u)));
}


public static Block
```

```
RandomBlock(Vector3int position, int r) {
 List < Vector3int > vectors = new List < Vector3int > ();
 for (int i = 0; i < 8 * r * r * r; ++i) // (2*radius)^3
  vectors.Add(
  new Vector3int(UnityEngine.Random.Range(-r, r),
   UnityEngine.Random.Range(-r, r),
   UnityEngine.Random.Range(-r, r)));
 return new Block(position, 1, vectors.ToArray());
}

public static Block
RandomStandardBlock(Vector3int position) {
 return new Block(position, standardBlocks[
  UnityEngine.Random.Range(0, standardBlocks.Count)]);
}
public static Block
LBlock(Vector3int position, int r) {
 List < Vector3int > vectors = new List < Vector3int > ();
 for (int i = -r; i <= r; ++i)
  vectors.Add(new Vector3int(0, 0, i));
 vectors.Add(new Vector3int(1, 0, r));
 vectors.Add(new Vector3int(0, 1, -r + 1));
 return new Block(position, 1, vectors.ToArray());
}



//Draw
public void Draw() {
 for (int i = 0; i < this.bricks.Count; ++i)
  this.AbsPos(bricks[i]).Draw(color);
}

public override string ToString() {
 string ret = "";
 ret += position + ";";
 bricks.ForEach(b => ret += b);
 return ret;
}

// baza standardnik blokova
public static List < Block > standardBlocks = new List <
   Block > {
 //dugi
```

```
new Block(Vector3int.Zero, 0,
 new Vector3int(0, 0, 0), new Vector3int(1, 0, 0),
 new Vector3int(-1, 0, 0), new Vector3int(2, 0, 0)),
// L blok
new Block(Vector3int.Zero, 1,
 new Vector3int(0, 0, 0), new Vector3int(1, 0, 0),
 new Vector3int(-1, 0, 0), new Vector3int(1, 0, 1)),
// pseudo L blok
/*   new Block( Vector3int.Zero,2,
                new Vector3int(0,0,0),  new
                   Vector3int(1,0,0),
                new Vector3int(-1,0,0), new
                   Vector3int(2,0,0),
                new Vector3int(1,0,1)),
   */ // T blok
new Block(Vector3int.Zero, 3,
 new Vector3int(0, 0, 0), new Vector3int(1, 0, 0),
 new Vector3int(-1, 0, 0), new Vector3int(0, 0, 1)),
// S blok
new Block(Vector3int.Zero, 4,
 new Vector3int(0, 0, 0), new Vector3int(0, 0, 1),
 new Vector3int(-1, 0, 0), new Vector3int(1, 0, 1)),
// kocka
new Block(Vector3int.Zero, 5,
 new Vector3int(0, 0, 0), new Vector3int(1, 0, 0),
 new Vector3int(0, 0, 1), new Vector3int(1, 0, 1),
 new Vector3int(0, 1, 0), new Vector3int(1, 1, 0),
 new Vector3int(0, 1, 1), new Vector3int(1, 1, 1)),

// puni
/*                   new Block( Vector3int.Zero,5,
                               new Vector3int(-1,0,0),
                                  new Vector3int(0,0,0),
                               new Vector3int(1,0,0), new
                                  Vector3int(-2,0,0),
                               new Vector3int(-1,0,1),
                                  new Vector3int(0,0,1),
                               new Vector3int(1,0,1), new
                                  Vector3int(-2,0,1),
                               new Vector3int(-1,0,-2),
                                  new Vector3int(0,0,-2),
                               new Vector3int(1,0,-2), new
                                  Vector3int(-2,0,-2),
                               new Vector3int(-1,0,-1),
                                             new Vector3int(0,0,-1),
                                          new Vector3int(1,0,-1), new
                                             Vector3int(-2,0,-1))
      */
   };

   public static void AddFullBlock(int n, int m) {
    List < Vector3int > vs = new List < Vector3int > ();

    for (int i = 0; i < n; ++i)
     for (int j = 0; j < m; ++j)
      if (i != 0 || j != 0) vs.Add(new Vector3int(i - n / 2,
         0, j - n / 2));

    standardBlocks.Add(new Block(Vector3int.Zero, 7,
       vs.ToArray()));
   }
  }
}
```

3.2 class Tetris

Klasa `Tetris` sadrži listu svih dosad spuštenih blokova koji su na sceni. Implementira mehaniku igre, funkcije `SpawnNextBlock()` za stvaranje sljedečeg bloka; `CheckSolved-Levels()` provjerava dali je razina popunjena i briše je, kao i nekolicinu funkcija za provjeru legalnosti poteza, te funkcije za procesuiranje inputa.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using UnityEngine;

using State = MainMenu.State;
using Controls = MainMenu.Controls;

namespace Tetris {
 class Tetris {
  public MainMenu mainMenu;

  public Vector3int dimensions; // sirina visina dubina
  public Vector3int spawnPosition;
  public List < Block > spawnedBlocks = new List < Block > ();
  public Block currentBlock;
  public Block nextBlock;

  float time = 0 f;
  int frq = 1;

  //za lakse iteriranje po XZ ravnini
  private List < int > xs = new List < int > ();
  private List < int > zs = new List < int > ();

  public int score;
  public int level = 0;
  public float speed = 0.7 f;

  public CameraScript cam1;
  public CameraScript cam2;

  public Tetris(Vector3int dimensions) {
   this.dimensions = dimensions;
   this.spawnPosition = new Vector3int(dimensions.X / 2,
      dimensions.Y - 2, dimensions.Z / 2);
   for (int i = 0; i < dimensions.X; ++i) xs.Add(i);
   for (int k = 0; k < dimensions.Z; ++k) zs.Add(k);
   //Block.AddFullBlock(dimensions.X,dimensions.Z);
  }

  // Game
  public void StartGame() {
   spawnedBlocks.Clear();
   nextBlock = null;
   SpawnNextBlock();
   score = 0;
   level = 0;
  }

  public void SpawnNextBlock() {
     //spawnedBlocks.Add(Block.RandomBlock(spawnPosition,2));
   if (nextBlock == null) nextBlock =
      Block.RandomStandardBlock(spawnPosition);
   spawnedBlocks.Add(nextBlock);
   nextBlock = Block.RandomStandardBlock(spawnPosition);
   currentBlock = spawnedBlocks.Last();
   frq = 1;
   if (InvalidPosition(currentBlock))
      mainMenu.NoResume(score);
  }

  public void UpdateGame() {
   time += Time.deltaTime;
   if (time > 1 / ((float) frq * (level + 1) * speed)) {
    time = 0 f;
    this.NextStep();
    SetToDraw();
   }
  }
  private void NextStep() {
   if (CanCurrentBlockMove(Vector3int.Down))
      currentBlock.MoveDown();
   else {
    CheckSolvedLevels();
    Sounds.sounds["DropDown"].Play();
    SpawnNextBlock();
   }
  }

  private void CheckSolvedLevels() {
   List < int > ys = currentBlock.bricks.Select(v =>
      currentBlock.AbsPos(v).Y).Distinct().ToList < int > ();
   ys.RemoveAll(y => xs.Any(x => zs.Any(z =>
    spawnedBlocks.All(block => block.bricks.All(v =>
     block.AbsPos(v) != new Vector3int(x, y, z))))));
   level += ys.Count * ys.Count;
   score = 100 * level;
   DeleteLevels(ys);
   LowerLevels(ys);
  }

  private void DeleteLevels(List < int > ys) {
   Sounds.sounds["DeleteLevel"].Play();
   spawnedBlocks.ForEach(block => block.bricks.RemoveAll(v =>
      ys.Contains(block.AbsPos(v).Y)));
  }
  private void LowerLevels(List < int > ys) {
   spawnedBlocks.ForEach(block => block.bricks.ForEach(v =>
    v.MoveDown(ys.Count(y => y < block.AbsPos(v).Y))));
  }

  // Movement,Rotation
  private bool
```

```
CanCurrentBlockMove(Vector3int direction) {
 return InvalidPosition(currentBlock.GetMoved(direction))
    == false;
}
private bool
CanCurrentBlockRotate(Vector3int rotation) {
 return InvalidPosition(currentBlock.GetRotated(rotation))
    == false;
}



// Collision
private bool IsOutOfStage(Block block) {
 return block.NotBoundedZeroAndUpper(dimensions);
}
private bool InvalidPosition(Block block) {
 return IsOutOfStage(block) || spawnedBlocks.Any(block2 =>
    block2 != currentBlock && Block.CollisionTest(block,
    block2) == true);
}


// Input
public void UpdateInput() {
 Vector3int dir = GetDirectionFromInput();
 Vector3int rot = GetRotationFromInput();
 if (rot != Vector3int.Zero) {
  if (CanCurrentBlockRotate(rot)) {
   currentBlock.Rotate(rot);
   Sounds.sounds["Rotation"].Play();
   SetToDraw();
  } else Sounds.sounds["InvalidPosition"].Play();
 }
 if (dir != Vector3int.Zero) {
  if (CanCurrentBlockMove(dir)) {
   currentBlock.Move(dir);
   SetToDraw();
  } else Sounds.sounds["InvalidPosition"].Play();
 }
 ResetVars();
}
```

```
// Draw
public void Draw() {
 spawnedBlocks.ForEach(block => block.Draw());
 Drawing.DrawAxis(dimensions);
}
public void DrawNextBlock() {
 nextBlock.GetMoved(new Vector3int(0, 50, 0) -
    spawnPosition).Draw();
}



static bool dirDownKey, dirUpKey, dirLeftKey, dirRightKey;

static int xInc, xDec, yInc, yDec, zInc, zDec;

static float scrWidth, scrHeight, btnSize;

Rect dirDownRect, dirUpRect, dirLeftRect, dirRightRect,
   xIncRect, xDecRect, yIncRect, yDecRect, zIncRect,
   zDecRect, dropDownRect;

void SetGUIVars() {
 scrWidth = Screen.width;
 scrHeight = Screen.height;
 btnSize = 0.15 f;

 dirDownRect = new Rect(0.775 f * scrWidth, scrHeight -
    0.15 f * scrWidth, btnSize * scrWidth, btnSize *
    scrWidth);
 dirUpRect = new Rect(0.775 f * scrWidth, scrHeight - 0.45
    f * scrWidth, btnSize * scrWidth, btnSize * scrWidth);
 dirLeftRect = new Rect(0.7 f * scrWidth, scrHeight - 0.3 f
    * scrWidth, btnSize * scrWidth, btnSize * scrWidth);
 dirRightRect = new Rect(0.85 f * scrWidth, scrHeight - 0.3
    f * scrWidth, btnSize * scrWidth, btnSize * scrWidth);


 xIncRect = new Rect(0.075 f * scrWidth, scrHeight - 0.45 f
    * scrWidth, btnSize * scrWidth, btnSize * scrWidth);
 xDecRect = new Rect(0.075 f * scrWidth, scrHeight - 0.15 f
    * scrWidth, btnSize * scrWidth, btnSize * scrWidth);
 yIncRect = new Rect(0.0 f * scrWidth, scrHeight - 0.3 f *
    scrWidth, btnSize * scrWidth, btnSize * scrWidth);
```

```
  yDecRect = new Rect(0.15 f * scrWidth, scrHeight - 0.3 f *
     scrWidth, btnSize * scrWidth, btnSize * scrWidth);
  zIncRect = new Rect(0.15 f * scrWidth, scrHeight - 0.6 f *
     scrWidth, btnSize * scrWidth, btnSize * scrWidth);
  zDecRect = new Rect(0.0 f * scrWidth, scrHeight - 0.6 f *
     scrWidth, btnSize * scrWidth, btnSize * scrWidth);

  dropDownRect = new Rect(0.35 f * scrWidth, scrHeight -
     0.15 f * scrWidth, 0.3 f * scrWidth, btnSize *
     scrWidth);

 }

 public void OnGUI() {
  GUI.skin = mainMenu.defaultSkin;
  SetGUIVars();

  switch (mainMenu.controls) {
   case Controls.Swipe:


    break;

   case Controls.Buttons:

    if (GUI.Button(dirDownRect, "",
       Styles.styles["dirdown"])) dirDownKey = true;
    if (GUI.Button(dirUpRect, "", Styles.styles["dirup"]))
       dirUpKey = true;
    if (GUI.Button(dirLeftRect, "",
       Styles.styles["dirleft"])) dirLeftKey = true;
    if (GUI.Button(dirRightRect, "",
       Styles.styles["dirright"])) dirRightKey = true;
    if (GUI.Button(xIncRect, "", Styles.styles["rotXLc"]))
       xInc = 1;
    if (GUI.Button(xDecRect, "", Styles.styles["rotXRc"]))
       xDec = 1;
    if (GUI.Button(yIncRect, "", Styles.styles["rotYLc"]))
       yInc = 1;
    if (GUI.Button(yDecRect, "", Styles.styles["rotYRc"]))
       yDec = 1;
    if (GUI.Button(zIncRect, "", Styles.styles["rotZLc"]))
       zInc = 1;
    if (GUI.Button(zDecRect, "", Styles.styles["rotZRc"]))
```

```
        zDec = 1;

      break;
    }

    if (GUI.Button(dropDownRect, "",
       Styles.styles["dropdown2"])) frq = (frq * 10) % 11;

    GUI.Label(new Rect(Screen.width * (1 f - 2 * btnSize),
       0.02 f * Screen.height, btnSize * scrWidth, btnSize *
       scrHeight), "score:" + score.ToString());
  }

  bool rotationMode = false;

  public void SetRotationMode(bool b) {
    rotationMode = b;
   }
   // pretplati se na static event iz MiniGestureRecognizera
  public void SetVarsFromSwipe(SwipeDetector.SwipeDirection
      dir) {
   if (mainMenu.controls == Controls.Swipe) {
     if (rotationMode) {
      if (dir == SwipeDetector.SwipeDirection.Up) xInc = 1;
      if (dir == SwipeDetector.SwipeDirection.Down) xDec = 1;
      if (dir == SwipeDetector.SwipeDirection.Left) yInc = 1;
      if (dir == SwipeDetector.SwipeDirection.Right) yDec = 1;
     } else {
      if (dir == SwipeDetector.SwipeDirection.Up) dirUpKey =
         true;
      if (dir == SwipeDetector.SwipeDirection.Down) dirDownKey
         = true;
      if (dir == SwipeDetector.SwipeDirection.Left) dirLeftKey
         = true;
      if (dir == SwipeDetector.SwipeDirection.Right)
         dirRightKey = true;
     }
   }
  }

  void ResetVars() {
   dirDownKey = false;
   dirUpKey = false;
   dirLeftKey = false;
```

```
      dirRightKey = false;
      xInc = 0;
      xDec = 0;
      yInc = 0;
      yDec = 0;
      zInc = 0;
      zDec = 0;
    }

    public static Vector3int GetDirectionFromInput() {
     return new Vector3int((dirRightKey ? 1 : 0) - (dirLeftKey
        ? 1 : 0),
      0,
      (dirUpKey ? 1 : 0) - (dirDownKey ? 1 : 0));
    }

    public static Vector3int GetRotationFromInput() {
     return new Vector3int((xInc - xDec) * (1 - yInc - yDec +
        yInc * yDec) * (1 - zInc - zDec + zInc * zDec),
      (yInc - yDec) * (1 - xInc - xDec + xInc * xDec) * (1 -
         zInc - zDec + zInc * zDec),
      (zInc - zDec) * (1 - xInc - xDec + xInc * xDec) * (1 -
         yInc - yDec + yInc * yDec));
    }


    private void SetToDraw() { // cam1.setToDraw();
        cam2.setToDraw(); Debug.Log("draw");
    }


  }
}
```

3.3 class TetrisMain

Ova klasa proširuje Unity baznu klasu MonoBehaviour i implementira funkcije za stvaranje i restart nove igre.

```
using UnityEngine;
using System.Collections;
using Tetris;
using System.Collections.Generic;
using Tetris;
using State = MainMenu.State;

// backgrounMusic source: Korobeiniki ,
   https://vgmdaily.wordpress.com/2009/10/22/korobeiniki-aka-tetris-type-
public class TetrisMain: MonoBehaviour {
  public MainMenu mainMenu;
  Tetris.Tetris tetris = new Tetris.Tetris(new Vector3int(6,
     18, 6));

  public void Set(MainMenu mainMenu) {
   this.mainMenu = mainMenu;
   tetris.mainMenu = mainMenu;
  }
  void Awake() {
   CreateLineMaterial();
   tetris.cam1 =
      transform.GetChild(0).GetChild(0).GetComponent <
      CameraScript > ();
   tetris.cam1.main = this;
   tetris.cam2 =
      transform.GetChild(1).GetChild(0).GetComponent <
      CameraScript > ();
   tetris.cam2.main = this;
   transform.GetChild(2).GetChild(0).GetComponent <
      NextBlockCamera > ().main = this;
  }
  public void Restart() {
   SwipeDetector.Swipe += tetris.SetVarsFromSwipe;
   SwipeDetector.RotationMode += tetris.SetRotationMode;
   tetris.StartGame();
   Sounds.sounds["backgroundMusic"].loop = true;
   Sounds.sounds["backgroundMusic"].Stop();
   Sounds.sounds["backgroundMusic"].Play();
  }

  void Update() {
   if (mainMenu.state == State.Game) {
    tetris.UpdateInput();
    tetris.UpdateGame();
   }
  }
  public void Draw() {
   lineMaterial.SetPass(0);
   tetris.Draw();
  }

  public void DrawNextBlock() {
   lineMaterial.SetPass(0);
   tetris.DrawNextBlock();
  }

  public void OnGUI() {
   if (mainMenu.state == State.Game) tetris.OnGUI();
  }
```

LITERATURA

[1] Unity Scripting Reference. http://docs.unity3d.com/ScriptReference/index.html