3DHERO: AN INTERACTIVE PUZZLE GAME PLATFORM FOR 3D SPATIAL AND REASONING TRAINING USING GAME ENGINE AND MACHINE LEARNING

David Tang[1] and Yu Sun[2]

[1]Irvine High School, 4321 Walnut Avenue, Irvine, CA 92604
[2]California State Polytechnic University,
Pomona, CA, 91768, Irvine, CA 92620

ABSTRACT

The well-known puzzle game Tetris, where arrangements of 4 squares (tetrominoes) fall onto the field like meteors, has been found to increase the brain's efficiency [1]. Many variations came into existence ever since its invention. Sometimes, the leveling can become a double-edged sword, so this game is essentially a Zen mode without a leveling system. This game is built for people who want to play a 3D version of Tetris at a speed they themselves have set. This paper designs a game to exercise spatial visualization. This study uses a Unity/C++-based game [2]. This game will be tested by kids on the autism spectrum, and we will conduct a qualitative evaluation of the approach.

No results have been shown yet, and that is due to the fact that this study is still a work in progress. I am trying to make the game comply with the latest Tetris design guidelines that I can find online (that is, the 2009 guideline).

KEYWORDS

Tetris, Spatial visualization, 3-Dimensional Perception.

1. INTRODUCTION

Tetris is a puzzle game created by the famous Soviet-American game developer Alexey Pajitnov and released in 1984 [3]. Developed in the Soviet Academy of Sciences in Moscow, Tetris was based on the famous pentomino puzzles Pajitnov liked to play with when he was a child [4]. He adapted the game to Cold War-era hardware and tweaked the game by reducing pentominoes to tetrominoes (hence the name, a portmanteau of "tennis", one of Pajitnov's favorite sports, and "tetra", meaning "four" in Latin) and creating a playing field where tetrominoes would fall like meteorites [5]. Pajitnov and his team realized that the game would end too quickly without a key feature: making rows disappear whenever players filled them up. People that Pajitnov had worked with were attracted to the game, and the game is still popular today, even leading some to make variants with special twists in them, including but not limited to "Not Tetris", with a physics engine and free-rotating tetrominoes; and a 3D version developed by T&E Soft for the Virtual Boy. It has inspired competitions to see who can earn the highest score. People have placed tetrominoes in specific arrangements at the beginning of their game to score more points.

David C. Wyld et al. (Eds): ARIA, SIPR, SOFEA, CSEN, DSML, NLP, EDTECH, NCWC - 2022
pp. 249-262, 2022. CS & IT - CSCP 2022 DOI: 10.5121/csit.2022.121520

Research from Mind Research Lab in Albuquerque on the original game has led to research on variations of the game, and its effects on autism [11]. This game incorporates features found in numerous other games before it. The only feature setting it apart from other games is a "central cube rotation system".

There already exists the aforementioned Virtual Boy version, and another game called Blockout. Though Blockout has an indication for the height of the playing field, it only provides a top view of the playing field [10]. However, this Unity remake has a full 3D view of the playing field, allowing people to strategize where their piece will land using the ghost piece.

The pre-existing Tetris research involves the original 2D game [6].

In this paper, we follow the same line of research by … Our goal is for players of this game to visualize arrangements of cubes spatially. Our method is inspired by Alexey Pajitnov's Tetris and some other 3D Tetris-based games. There are some good features of Unity and C++. First, Unity is the second-most used game engine on Steam products. Second, we added more and more Unity plugins to the game.

The differences between my method and the other Tetris research project are that this paper focuses on Tetris as it relates to autism, and that this project uses an unofficial self-made 3D version.

The rest of the paper is organized as follows: Section 2 gives the details on the challenges that we met during the experiment and designing the sample; Section 3 focuses on the details of our solutions corresponding to the challenges that we mentioned in Section 2; Section 4 presents the relevant details about the experiment we did, following by presenting the related work in Section 5. Finally, Section 6 gives the conclusion remarks, as well as pointing out the future work of this project.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Implementing the Ghost Piece

The Ghost piece is a prediction of the landing position of a Tetromino if allowed to drop into the playing field. It is intended to reduce misdrops, especially for beginners and high-speed players.

According to the Tetris Wiki, the Ghost piece, or ghost for short, also called shadow or (in Arika games) Temporary Landing System (TLS), is a representation of where a tetromino or other piece will land if allowed to drop into the playfield [7]. It is generally colored fainter than the falling piece and the blocks in the playfield. As the player moves the falling piece, the ghost piece moves below it; when the piece falls far enough that it overlaps the ghost piece, the falling piece is always drawn in front. Older games did not have a ghost piece, but all games that conform to the Tetris Guideline allow the player to use a ghost piece at all times, and Dr. Mario for Nintendo 64 has a ghost piece as well. The ghost piece reduces the number of misdrops, especially for beginners or for high-speed players who use hard drop, but some players who are migrating from games without a ghost piece have trouble adjusting to the ghost piece when they fail to distinguish it from blocks in the playfield.

2.2. The Free-Rotating Camera

Blockout's camera is not free-rotating, and only gives a top view of the playing field. If implemented, it may block players' vision of the space below overhang. Instead, this Unity game features a plugin called Cinemachine. The camera rotates around an orbit point (which in this case refers to the center of the playing field)

3. SOLUTION

All scripts for this 3D Tetris game (working title) were coded in C++ [8].

The game is set on a black background with a white floor. Like in 3DT and Blockout, the goal is to clear planes [9]. Since the tetracubes rotate around one singular mino, T-Spin singles and doubles are possible, but not T-Spin triples and mini T-spins.

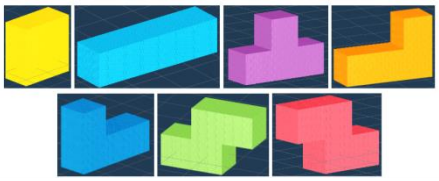

Figure 1. For the seven one-sided tetrominoes, I kept the colors for their tetracube counterparts. Top (left to right): O, I, T, and L. Bottom (left to right): J, S, and Z. L and J are chiral pairs in 2D, but not in 3D. Same applies to S and Z

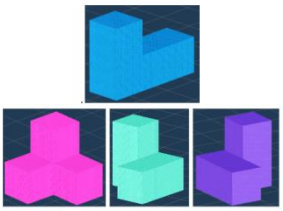

Figure 2. Left to right: The tripod, and the 3D chiral pair [left arm and right arm]. Those are new to the game because they do not exist in 2D

To create these arrangements, start with one cube, then clone it three times and move the clones until the shape is made.

Shown below is the code of the current tetracube spawner.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Events;
using System;
using Random = UnityEngine.Random;
using UnityEngine.InputSystem;

public class RandomPieceSpawner : MonoBehaviour
{
   [Header("Internal Info")]
```

```
    [SerializeField] int current = 0;
    [SerializeField] bool isRandomized = true;
    public List<TetraminoGeneratorUpdater> piecesList = new();
    [SerializeField] float waitTimer = 0.5f;
    [SerializeField, Tooltip("The amount to divide speed by to make the timer not decrease as
fast.")]
    float speedInverseMultiplier = 2;
    [SerializeField] float minimumWaitTime = 0.5f;

    [Header("Extrenal Info")]
    public GameObject piecesParent = null;
    public TetraminoGenerator tetGen;

    public TetraPieceScript currentPiece = null;

    public GameObject hintPrefab;
    // public PieceKeyboardControl defaultControl = new();
    [SerializeField] InputActionAsset controls;
    [Header("Events")]
    public UnityEvent OnGameOver = new ();
    public UnityEvent OnPieceSpawn = new ();

    // Private Variables
    private bool gameEnded = false;

    void Start()
    {
        Random.InitState(Random.Range(Int32.MinValue, Int32.MaxValue));
        if(piecesList.Count == 0)
        {
            Debug.LogError("Pieces List is Empty!");
            return;
        }
        tetGen?.UpdateGenerator(piecesList[current]);
    }


    // Update is called once per frame
    void Update()
    {
        if (gameEnded) return;

        if(currentPiece == null || currentPiece.CheckIfStopped)
        {
            SpawnPiece();
            OnPieceSpawn?.Invoke();
        }

    }

    public int RandomIndex()
    {
        return Random.Range(0,piecesList.Count);
    }

    public void SpawnPiece()
```

```
    {
        current = isRandomized ? RandomIndex(): current;
        tetGen?.UpdateGenerator(piecesList[current]);

        var piece = tetGen?.GenerateTetramino();
        piece.transform.position = Tetra3DGrid.ForceIntoGridPosition(transform.position);
        if(!Tetra3DGrid.CheckMovementOK(piece.transform.position, Vector3.zero))
        {
            gameEnded = true;
            currentPiece = null;
            OnGameOver?.Invoke();
            return;
        }

        piece.transform.parent = piecesParent != null ? piecesParent.transform : this.transform;

        Rigidbody rb = piece.AddComponent<Rigidbody>();
        rb.useGravity = false;
        rb.isKinematic = true;

        TetraPieceScript pieceScript = piece.AddComponent<TetraPieceScript>();
        // pieceScript.OverrideControls(defaultControl);
        currentPiece = pieceScript;
        pieceScript.SetControls(controls);
        pieceScript.OverrideTimings(newDropTimer: waitTimer);

        pieceScript.OnUnableToRegister.AddListener(this.GameOver);

        TetraPieceHints pieceHint = piece.AddComponent<TetraPieceHints>();
        pieceHint.pieceRef = pieceScript;
        pieceHint.hintPrefab = hintPrefab;

    }

    public void GameOver()
    {
        OnGameOver?.Invoke();
    }
    // Increase speed by decreasing waitTime
    public void IncreaseSpeed(float increaseAmount){
        waitTimer -= increaseAmount/speedInverseMultiplier;
        waitTimer = Mathf.Max(waitTimer, minimumWaitTime);
    }

    // Set wait timer by level
    public void SetSpeedByLevel(int level){
        waitTimer = Mathf.Pow((0.8f - ( (level - 1) * 0.007f) ), level - 1);
    }

    public void SetWaitTimer(float timerAmount){
        waitTimer =  Mathf.Max(timerAmount, minimumWaitTime);
    }
}
```

This segment of code always returns a random integer from 0 to the length of the piece list. Problem is, this algorithm generates piece droughts.

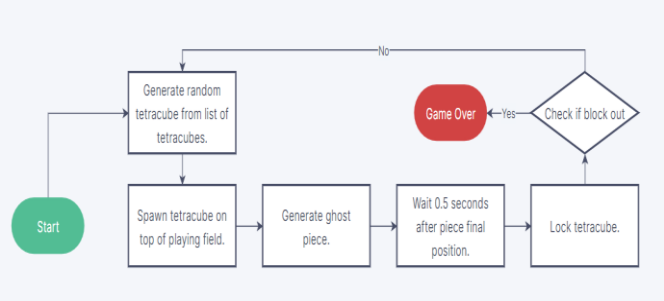

Figure 3. A flow chart describing how the randomizer and spawner in RandomPieceSpawner.cs works

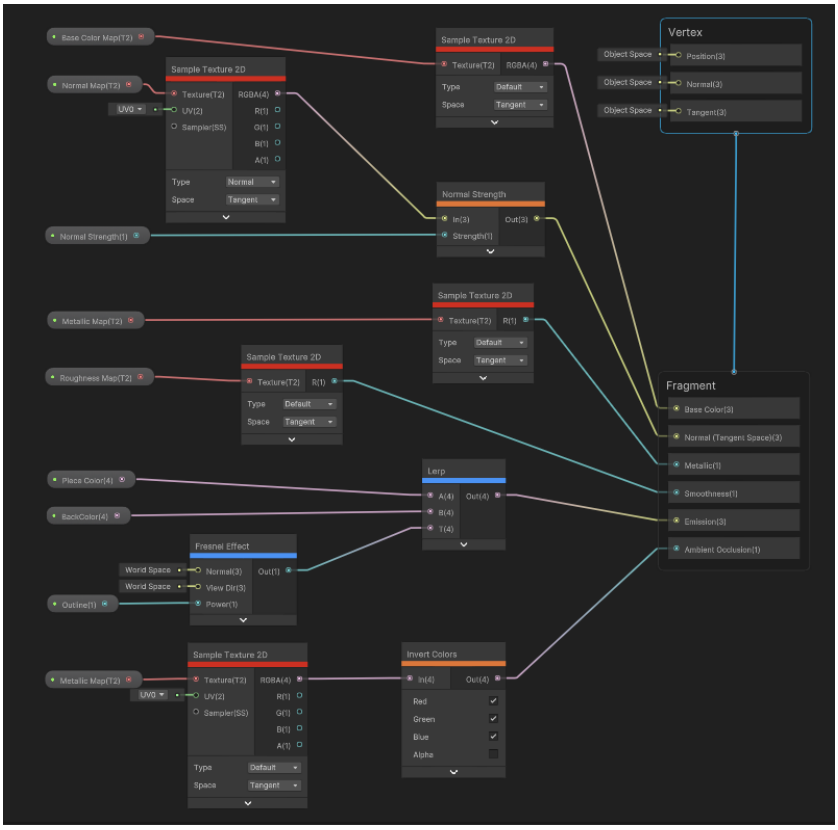

Figure 4. The Shader Graph for the Tetracubes

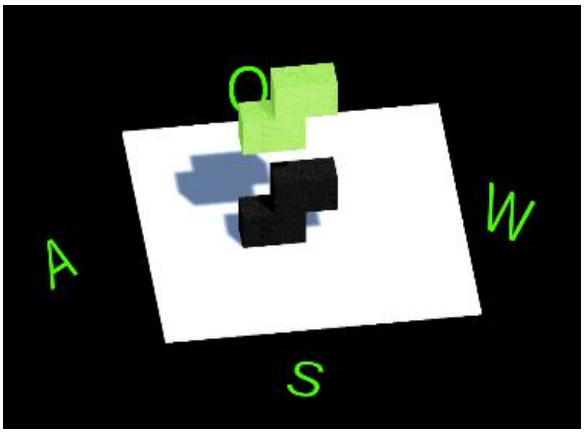

Figure 5. The S-tetracube, otherwise known as the N or the Z, with the shader as mentioned in Figure 4. Below the actual S-tetracube (in green) is a ghost piece (in black). The gray shadows are from the light source

Unlike in BlockOut, however, the camera is free-rotating, allowing players to view a full 3D view.

The code for the camera is as follows, and consists of two scripts: CameraFixedRotator.cs, which handles the rotation of the camera around a singular point; and CameraOrbitControls.cs, which handles the controls required to rotate the camera.

(CameraFixedRotator.cs)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.InputSystem;

public class CameraFixedRotater : MonoBehaviour
{
   public Key cwRotationKey = Key.Comma;
   public Key acwRotationKey = Key.Period;

   [SerializeField]
   private int xRemain = 1;
   [SerializeField]
   private int zRemain = -1;


   // Update is called once per frame
   void Update()
   {
      if(Keyboard.current[cwRotationKey].wasPressedThisFrame)
      {
         transform.rotation = Quaternion.Euler(transform.eulerAngles + Vector3.up*90);
         transform.position = new Vector3(transform.position.x * xRemain, transform.position.y,
transform.position.z * zRemain);
         xRemain *= -1;
         zRemain *= -1;
      }
      else if(Keyboard.current[acwRotationKey].wasPressedThisFrame)
      {
         transform.rotation = Quaternion.Euler(transform.eulerAngles + Vector3.down*90);
         transform.position = new Vector3(transform.position.x * zRemain, transform.position.y,
transform.position.z * xRemain);
         xRemain *= -1;
      zRemain *= -1;
      }
   }
}
```

```
(CameraOrbitControls.cs)
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.InputSystem;
using Cinemachine;

public class CameraOrbitControls : MonoBehaviour
{
   [SerializeField] CinemachineFreeLook orbitCamera;
   [SerializeField] float orbitSpeed = 1f;
   [SerializeField] bool invertYValue = false;

   private void Awake()
   {
      if(orbitCamera == null){
         orbitCamera = GetComponent<CinemachineFreeLook>();
      }
   }

   public void OnOrbitMove(InputAction.CallbackContext context)
   {
      Vector2 rotation = context.ReadValue<Vector2>().normalized;

      rotation.y = invertYValue ? -rotation.y : rotation.y;
      rotation.x = rotation.x * 180;

      orbitCamera.m_XAxis.Value = rotation.x * orbitSpeed * Time.deltaTime;
      orbitCamera.m_YAxis.Value = rotation.y * orbitSpeed * Time.deltaTime;
   }
}
```

The ghost piece is integrated into the game as a script called TetraPieceHints.cs.

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

public class TetraPieceHints : MonoBehaviour
{
   public float transparency = 0.25f;
   public TetraPieceScript pieceRef;
   public GameObject hintPrefab;

   private List<GameObject> hintHolder = new List<GameObject>();
   private bool isStopped = false;
   void Start()
   {
      pieceRef = GetComponent<TetraPieceScript>();
      foreach(Transform child in transform)
      {
         hintHolder.Add(Instantiate(hintPrefab, child));
      }
   }

   void Update()
   {
      if(isStopped) return;

      isStopped = pieceRef.CheckIfStopped;
      if(!isStopped && pieceRef != null && hintHolder.Count > 0)
      {
         UpdateAllPieces();
      }
      if(isStopped)
      {
         foreach(GameObject child in hintHolder)
         {
            Destroy(child);
         }
         hintHolder.Clear();
      }
   }

   public void UpdateAllPieces()
   {
      for(int moveAmount  = Tetra3DGrid.gridHeight+1; moveAmount > 0; moveAmount--)
      {
         bool allWorks = true;
         foreach (Transform child in this.transform)
         {
            if(!Tetra3DGrid.CheckMovementOK(child.position, Vector3.down*moveAmount))
            {
               allWorks = false;
               break;
            }
         }
         if(allWorks)
         {
            foreach(GameObject hint in hintHolder)
```

```
            {
               hint.transform.position = (Vector3.down * moveAmount)
                                    + hint.transform.parent.position;
            }
            break;
         }
      }
   }

   public Vector3 FindCollisionPoint(Transform child)
   {
      Vector3 direction = Vector3.down;
      Vector3 highestHit = Vector3.negativeInfinity;
      foreach(var collision in Physics.RaycastAll(child.position, direction,
Tetra3DGrid.gridHeight*2))
      {
         if(collision.collider.GetComponentInParent<TetraPieceScript>() != pieceRef
            || collision.collider.CompareTag("Finish") )
            {
               if(Mathf.Ceil(collision.point.y) > highestHit.y)
               {
                  highestHit = collision.point;
               }
            }
      }
      return highestHit;
   }
}
```

4. EXPERIMENT

4.1. Experiment 1

To test whether our games are really useful in helping children with autism focus, we found 9 children with autism of various ages from California. Divide them into 3 different groups. We separately calculated the playtime of children of different ages when playing this game, It was found that after a period of practice, children's attention time was significantly longer when playing the game. The results show that the game is most effective for 7-10-year-olds。7-10-year-olds can only play the game continuously for 15.1 seconds at first, After a period of time, it can reach more than 30 seconds.

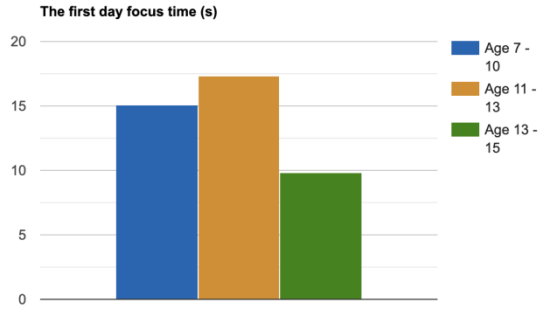

Figure 6. The first day focus time graph

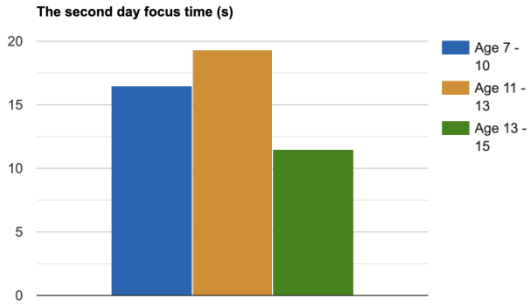

Figure 7. The second day focus time graph

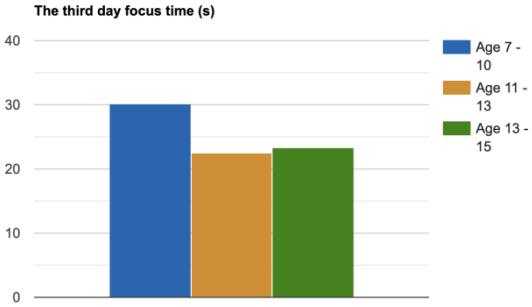

Figure 10. Result of experiment 2

5. RELATED WORK

The idea of a 3D version of Tetris wasn't new. There exists a Microsoft DOS game called Blockout, which implements polycubes of various orders, a ghost piece, and a top-view camera [12]. This project is similar to Blockout in the way that there is no next queue and that it also has the ghost piece implemented. The size of the playing field, and also the set of polycubes used, can be chosen by the player.

However, this project restricts the pieces available to the 8 one-sided tetracubes, and the camera is free-rotating. Also, the pieces are not colored based on stack height in this incarnation. They are rather colored based on what shape they are and what orientation they spawn in.

On to 3D Tetris, the Virtual Boy port by T&E Soft. The polycubes are only red due to the Virtual Boy only being able to display in black and red [13]. It, like Blockout, involves polycubes of orders 1-5, but also includes arrangements of pseudo-polyominoes extruded by 1 unit, and even arrangements where the cubes themselves are not connected at all. Whenever players max out, the bottom-most row gets cut and the playing field gets shorter. There is a layer-by-layer view of the playing field in 3D Tetris.

This project, however, has an 8x12x8 playing field in contrast to 3D Tetris' 5x5x5 playing field, and stays true to its title by restricting the pieces to the 8 one-sided tetracubes.

6. CONCLUSIONS

This paper talks about a 3D remake of Tetris (and by extension, BlockOut) while trying to adhere to the design guidelines as much as possible. It also involved elements from the already-existing Virtual Boy game. The game teaches people how to deal with piece droughts.

Interestingly, there exists a maxout TAS of NES Tetris without any I pieces [14].

The game was made by the Tetris design guidelines of 2009. There is no next queue. There is no visible grid. There is no soft dropping. Hard dropping does not generate the next piece instantly.

As for the future work, we plan to do or add the following: a warning system that warns players whenever they are about to "top out"; an awards system; T-spins and Mini T-spins; and a symmetrical SRS rotation system [15].

We are currently in the process of implementing these features into the game.
Here is a segment of pseudocode for the Random Generator and the Next queue, though they have not been implemented into the game itself.

- Make the entire next queue an ArrayList of tetracubes.
- Generate a random permutation of the bag of tetra cubes and add them to the next queue once there are (visible length of next queue) tetracubes left in the next queue.

REFERENCES

[1] Demaine, Erik D., Susan Hohenberger, and David Liben-Nowell. "Tetris is hard, even to approximate." International Computing and Combinatorics Conference. Springer, Berlin, Heidelberg, 2003.
[2] Mattingly, William A., et al. "Robot design using Unity for computer games and robotic simulations." 2012 17th International Conference on Computer Games (CGAMES). IEEE, 2012.
[3] Flom, Landon, and Cliff Robinson. "Using a genetic algorithm to weight an evaluation function for Tetris." Colorado State University, Tech. Rep. Luke (2004).
[4] Fortescue, Stephen. "The Russian Academy of sciences and the Soviet Academy of sciences: Continuity or disjunction?." Minerva 30.4 (1992): 459-478.
[5] Sears, Derek WG, and Robert T. Dodd. "Overview and classification of meteorites." Meteorites and the early solar system (1988): 3-31.
[6] Ak, Oguz, and Birgul Kutlu. "Comparing 2D and 3D game-based learning environments in terms of learning gains and student perceptions." British Journal of Educational Technology 48.1 (2017): 129-144.
[7] Zhao, Tianqu, and Hong Jiang. "Landing system for AR. Drone 2.0 using onboard camera and ROS." 2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC). IEEE, 2016.
[8] Nitsche, Michael. Video game spaces: image, play, and structure in 3D worlds. MIT Press, 2008.

[9] Murdock, Calvin, et al. "Blockout: Dynamic model selection for hierarchical deep networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
[10] Prabhakar, Balaji, Nick McKeown, and Jean Mairesse. "Tetris models for multicast switches." Proc. of the 30th Annual Conference on Information Sciences and Systems, Princeton. 1996.
[11] Jackson, Wallace. "Implementing Game Audio Assets: Using the JavaFX AudioClip Class Audio Sequencing Engine." Beginning Java 8 Games Development. Apress, Berkeley, CA, 2014. 323-341.
[12] Agah, Afrand, and Sajal K. Das. "Preventing DoS attacks in wireless sensor networks: A repeated game theory approach." Int. J. Netw. Secur. 5.2 (2007): 145-153.
[13] Li, Yuzhe, et al. "SINR-based DoS attack on remote state estimation: A game-theoretic approach." IEEE Transactions on Control of Network Systems 4.3 (2016): 632-642.
[14] Gibbons, William. "Blip, bloop, Bach? Some uses of classical music on the Nintendo entertainment system." Music and the Moving Image 2.1 (2009): 40-52.
[15] Bourne, S., and P. Bruggen. "Examination of the Distinctive Awards System." Br Med J 1.5950 (1975): 162-165.