A Way of Exploring the Real World Using Virtuality with the Example of Creating Animation in the Unity Environment

L. A. Khaliullina(✉) and I. I. Bikmullina

Kazan National Research Technical University Named After A.N. Tupolev, 10, St. Karl Marx, Kazan 420111, Russia
liya.khaliullina@gmail.com

Abstract. This article describes the method of creating animation in the Unity environment as one of the possible ways to study the real world and simulate physical processes, which would give an optimal balance between the achieved result and the resources used. The result of the work done is a generated hybrid method for creating a cartoon based on combining animations performed in several techniques. The uniqueness and novelty of the proposed method lies in the synergy of various tools for creating products in the Unity environment that meet the criteria for effective animation. It is worth noting the simplicity and speed of implementation, the clarity and accessibility of the presentation of information, which make it possible to use this method for educational purposes when teaching students of any age to create animation and awaken interest in the production of cartoons.

Keywords: Unity · Animation · Image · Animation · Technology

1 Introduction

In connection with the development of modern technologies, the need for rapid organizational changes in the conditions of a huge flow of incoming information and large amounts of data, there is an acute issue of training people and automating this process.

The learning process can be viewed as a joint activity of participants in organizing the assimilation of an academic discipline, obtaining subjectively new or new subject and procedural knowledge, and exploring the real world [1]. Certain requirements are imposed on the learning process: quick adaptation to changes, flexible management taking into account various teaching methods, constant interaction of the participants in the learning process, as well as creating opportunities for creativity, reflection, for using the subjectivity of the participants in the process [2]. Automation of training allows you to improve, maintain and control the skill level of training participants.

It is advisable to consider the learning process on the example of educational work carried out with a child using animation and animation, because the fundamental foundations of knowledge are laid in a person in childhood.

A. A. Radionov and V. R. Gasiyarov (Eds.): RusAutoCon 2022, LNEE 986, pp. 107–118, 2023.
https://doi.org/10.1007/978-3-031-22311-2_11

2 Relevance

A person receives more than 70% of information from the world around him through his eyes, perceives it with the help of images, therefore, in the early stages of development, one of the ways to cognize the surrounding reality is to watch cartoons that reflect the model of reality [3].

Cartoons have characteristics that give reason to consider them as a learning tool. Due to its accessibility and clarity for understanding, animation takes a strong position in learning issues and allows you to make the educational process the most interesting, to visualize it. That is why the need to find optimal programs for creating cartoons and finding solutions to simplify their development is an urgent problem for improving the learning process with the possibility of its further automation.

From the point of view of modern development of science and technology, we are observing the evolution of figurative means. The use of the latest technologies in animation had a beneficial effect on it, accelerating the process of working on works, raising their quality level, which contributed to the emergence of new means of expression, new animation forms [4]. Technologies have opened up new ways of influencing the viewer's consciousness through spatial sound, stereoscopic images, and three-dimensional graphics.

At the same time, one of the main values of animation is the versatility of the language, which makes it possible to organize a comprehensive system of integrated developmental education for children.

3 Formulation of the Problem

To organize high-quality educational work with the possibility of further automating the learning process, it was decided to propose a method for creating animation in the Unity environment, which would give an optimal balance between the achieved result and the resources used. Subsequently, the resulting animations can be used to create a full-fledged training animation video (cartoon).

It is necessary to offer the final version of the cartoon, which consists of two parts, which makes it possible to implement all stages of development, including the installation of the finished project. In the first part of the cartoon, the movement of a boat on the surface of the water should be presented, taking into account the waves, in the second part of the cartoon, a scene of the abduction of livestock by an unidentified flying object will be proposed, taking into account sound and voice acting, creating a background and scenery. The result of the work done will be a generated hybrid method for creating a cartoon based on combining animations performed in several techniques.

4 Analysis of Existing Technologies for the Development of Cartoons in the Unity Environment

Animation is a type of multimedia technology.

The word animation comes from the Latin root Anima. Animation is the mobility of an image.

Animation is a technology that allows you to create the illusion of movement using inanimate objects. The most popular form is animation, which is a series of hand-drawn images.

Animation is a technique for creating the illusion of moving images using a sequence of still images (frames) replacing each other with a certain frequency. Animated films are a kind of feature films.

There are several types of animation, which, according to the method of creation, are subdivided into:

1) Plasticine animation - a type of animation, where the film is made by time-lapse shooting of plasticine objects, with their modification in the intervals between the frames taken.
2) Hand-drawn animation - animation technology based on time-lapse shooting of slightly different two-dimensional drawings.
3) Puppet animation - a method of volumetric animation. When creating, a scene-layout and doll-actors are used. The scene is photographed frame by frame, after each frame minimal changes are made to the scene (for example, the pose of the doll changes). When the resulting sequence of frames is played back, the illusion of movement of objects appears.
4) Computer animation - sequential display (slideshow) of previously prepared graphic files, as well as computer simulation of movement by changing (and redrawing) the shape of objects or showing sequential images with phases of movement.
5) Sand animation - an animation technology in which a light powder (usually cleaned and sifted sand, but also salt, coffee, or something similar) is applied in thin layers to glass and mixed, creating a moving picture (usually all actions are performed by hand, but brushes can also be used as accessories). The resulting image can be transmitted to the screen using an overhead projector or light board.

By duration

1. Feature-length cartoons: cartoons usually over 70 min in length.
2. Short cartoons (usually about 25–30 min).

Cartoons created using computer graphics are divided by type:

4.1 Animating a Rigid Body

Solid animation is used to create ready-made animation sequences that move or change the properties of objects, taking into account the fact that the objects are a single whole, as opposed to objects moving relative to the objects themselves in the part. Some examples of this type of animation are the movement of a racing car on the road, a hinged door opening, a spaceship flying through its trajectory space, and a piano falling from a building window. Despite the differences between the examples, they all have an important property. In particular, although changes are made to keyframes, they are made for a single and cohesive object. In other words, although the door hinges from closed to open, it still ends up as a door, with the same internal structure and the same structure as

before. She does not transform into a tiger or a lion. It doesn't explode or jelly. It does not dissolve in raindrops. During the animation, the door retains its physical portability. It changes only in terms of its position, speed and scale. Thus, when animating a solid, changes to keyframes propagate only whole objects and their high-level properties. They are not related to subproperties and internal components. These kinds of animations can either be imported directly in the Unity Animation Editor, or in 3D animation programs (such as Maya, Max, or Blender) and then imported into Unity.

4.2 Animation with Sprites

Sprite animation is one of those things that, for all their primitiveness, have been successfully working and applied in computer graphics and games for more than a quarter of a century. Even in 3D games there are sprites - for example, explosion billboards. In many browser and flash games, sprite animation is used, since it is very simple and does not require high performance.

If one sprite is a still picture, then a series of these pictures quickly replacing each other makes up an animation called a sprite. This type of animation differs in that it is not the entire frame (frame) that changes on the screen, but only its small piece, where the sprites appear. Also, sprite animation is sometimes called software animation.

Figure 1 shows an example of the sprite animation.

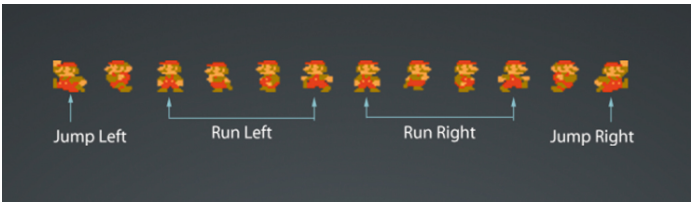

Fig. 1. Example of the sprite animation.

4.3 Animation Using Scripts

Scripting is an essential part of all games. Even the simplest games need scripts for player reactions and gameplay events. In addition, scripts can be used to create graphical effects, control the physical behavior of objects, or implement a custom AI system for game characters.

An example script is presented below:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Wave_Controller: MonoBehaviour.
{ public float height;
public float time;
// Start is called before the first frame update.
```

```
void Start().
{
iTween.MoveBy(this.gameObject, iTween.Hash("y", height, "time", time, "loop-
type", "pingpong", "easet0ype", iTween.EaseType.easeInOutSine));
}.
// Update is called once per frame.
void Update().
{}}.
```

4.4 Rigged Animation, or Skeleton Based Animation

Skeletal animation is a way to animate 3D models in animation and computer games. It consists in the fact that the multiplier creates a skeleton, which is usually a tree-like structure of bones, in which each subsequent bone is "tied" to the previous one, that is, it repeats movements and turns after it, taking into account the hierarchy in the skeleton. Further, each vertex of the model is "snapped" to some bone of the skeleton. Thus, when a single bone moves, all vertices attached to it move as well. Thanks to this, the animator's task is greatly simplified, because there is no need to animate each vertex of the model separately, and it is enough to set the position and rotation of the bones of the skeleton.

Also, thanks to this method, the amount of information that is needed for animation is also reduced. It is enough to store information about the movement of bones, and the movements of the vertices are calculated based on them.

Figure 2 shows an example of a skeletal animation.

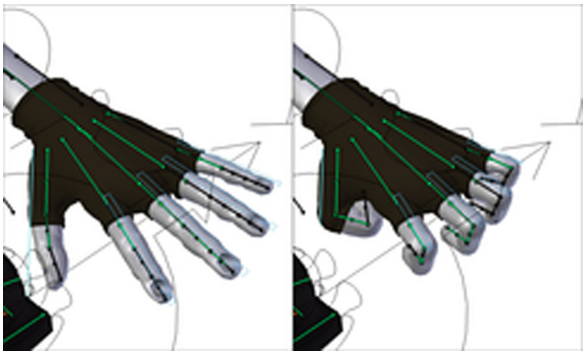

Fig. 2. Example of a skeletal animation.

Skeletal animation with weights of bones is a more advanced version of skeletal animation, in which each vertex of the model can be associated with not one, but several bones. In this case, for each bone, its weight is determined, that is, the magnitude of the influence of this bone on the displacement of the vertex. The greater the weight of the bone, the more the vertex moves under its influence. Thanks to the distribution of scales, you can animate smooth bends of surfaces, the movement of fabrics, flags.

Skeletal animation as a technology is also used in two-dimensional computer multiplication, for example, in the vector multiplication editor Anime Studio or the open Synfig Studio. Separate fragments of the model are "tied" to the bones, and then the multiplier sets the movement of the bones. The character model moves, following the movements of the skeleton.

5 Stages of Development of Cartoons

Usually, the development of a full-length cartoon includes several stages:

1) script writing;
2) determining the way to create a cartoon;
3) choosing a program for creating animation;
4) cartoon storyboard;
5) creating backgrounds and decorations;
6) soundtrack and voice acting;
7) installation of the finished project.

In order to understand the basic principles of cartoon development, we will consider the creation of computer animation using the example of the Unity game engine using various technologies.

Technology is a set and sequence of methods and processes for converting raw materials, which make it possible to obtain products with specified parameters [5].

Methodology is a field of knowledge that studies the means, prerequisites and principles of organizing cognitive and practical-transformative activities, the main tool is the method [6].

The term "educational animation" means any moving image that is used for educational purposes [7]. It can be just a moving arrow on a diagram or a full-fledged cartoon on your topic. The main reason many teachers create animations is to improve teaching efficiency.

Animation allows you to visually explain the content and meaning of complex concepts, abstract concepts. Information on a variety of topics can be presented in the form of animated images, and it is suitable for students of any age.

When adding animation effects, it is important to know some rules, because low-quality animation can alienate the viewer. Effective animation must have a number of characteristics:

- harmonious combination of text and images;
- not too much information;
- easy-to-read playback speed;
- emphasis on the most important elements;
- interactivity (the ability to pause, rewind, change the image scale, control the playback speed);
- voice guidance [8].

Difficulties may arise at each stage of the development of a full-length cartoon. In particular, the Unity3D game engine is suitable for the implementation of the cartoon. It has a number of advantages, such as cross-platform, good Community, which means that the various functions of the engine have a clear description with examples on the developer's website, which you can refer to at any time, and the Asset Store, where there are a huge number of different plugins and resources for creation of cartoons and games.

The Unity editor has ports for OS X and Windows, and was originally designed for OS X. Unity includes support for DirectX 11, which opens the way for your applications to the worlds of Windows 8 and Windows Phone 8. The Unity engine is especially valuable for its low barriers to entry for novice users, free indie version. The low barrier to entry is the result of well-designed application. Many things can be done using various editors without writing a single line of code (the code is written in JavaScript, C #, Boo). The C / C + + source code is closed, but this does not pose any obstacles due to the extended component structure of the Unity engine.

However, when analyzing technologies for the development of cartoons in the Unity environment, we faced the problem of the lack of clear algorithms for creating cartoons, there are only separate methods tied to specific types of animation. Therefore, in order to get the technology for creating cartoons, you need to develop your own algorithms and methods for the successful implementation of your cartoon.

6 Realization

To achieve the set goals in scientific work, it is necessary to use various methods, however, the problem of this study is that during the detailed analysis, it was revealed that there were no clearly formulated methods for developing cartoons in the Unity environment and ready-made practical solutions with detailed instructions. As a result, it became necessary to describe our own method as one of the possible ways to develop cartoons, which would give the optimal ratio between the achieved result and the resources used.

The described method is a synergy of various tools for creating products in the Unity environment that meet the criteria for effective animation.

Any development starts with creating an empty project. We place a Plane named "Water_Plane" and parameters in the Scale column $x = 5$, $y = 5$, $z = 5$. Then we add the Cloth component. It is a physical tissue modeling solution that works in conjunction with the Skinned Mesh Renderer technique for rendering bone animations [9].

You need to remove the Mesh Collider component that controls the main Unity graphics primitive (3D meshes) and select Plane in the Mesh column of the Skinned Mesh Renderer component.

You must select Edit cloth constraints in the Cloth component. The Stretching Stiffness value changes by 0.01.

In the next step, we hold down the Shift button and the left mouse button, select the Plane border, check the Max Distance checkbox and set the value to 0.2. These actions are repeated for all four boundaries (to check: the selected points should change their color).

It is necessary to uncheck the box opposite the Use Gravity column of the Cloth component. Then we download the iTween module package from the Asset Store and import it by unchecking the Sample box.

We add a script called "Wave_Controller" to our plane.

We set the values for height = 2.67 and time = 2.35 for our waves. It is necessary to change the value of the World Acceleration Scale of the Cloth component to 0.85, add a wave texture to the Assets and transfer it to Plane.

Change the values of the Tilling graph of the Wave Texture component to x = 5 and y = 5 and the Shader graph of the Wave Texture component to Legacy Shaders / Diffuse.

Download the Old Wooden Row Boat v2 unit package from the Asset Store and import it. Next, transfer the boat to Plane. Change the values for the boat to x = 1, y = 1, z = 1. Create an empty object named Boat and transfer Old_Boat_Prefab into it [10].

Next, we create the BoatFloater script. Change the name of Plane to Sea and add the Boat object to it. By independently changing the values of the Stretching Stiffness, World Acceleration Scale, height, time parameters, you can track how the waves change.

Next, install the Rock Package and import it. Add several stones to the Plane, changing their position and size.

This concludes the first part of the development of our cartoon. Figure 3 shows an example of the first computer animation using the Unity game engine.

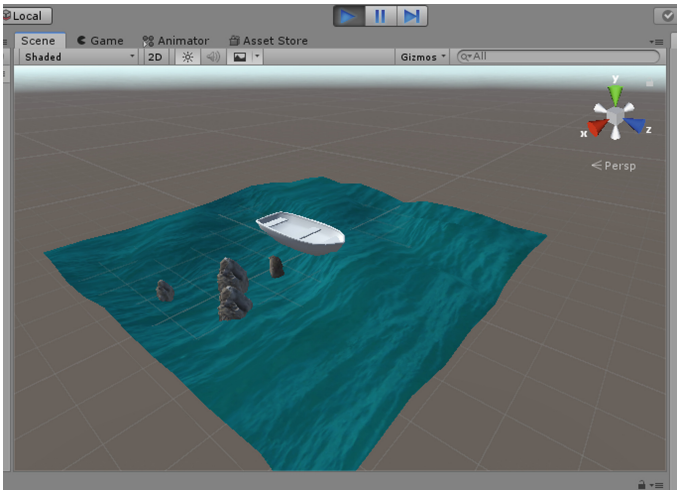

Fig. 3. Example of the first computer animation using the Unity game engine.

The second part of the development of our cartoon is the capture of cows using a teleportation beam before they reach a safe barn. We will create it as a 2D project, which at the installation stage can be connected to the first part [11].

We select the required platform in the Build Settings section of the File menu. Next, you need to select the size of the pictures that we will use. Before we start coding, we need to add the Asset to the Unity project. There are several ways to do this:

1. select Import New Asset from the Assets menu;
2. add details to the assets folder in the project;
3. drag the parts to the assets folder in the project.

After completing this step, we will see the assets of our project in the Assets folder in the Project panel [12].

Next, we create a scene for our animation by dragging and dropping objects into the Hierarchy or Scene panel. Then you need to transfer the background to the Hierarchy panel. We can select the Main camera in the Hierarchy panel and see what the camera will display. We will change the Size parameter for the Main Camera to 1.6 in the Inspector to see the whole scene.

The ship in our game is a static element and the player does not interact with it in any way. We'll position it in the center of the scene by dragging the aircraft from the Assets folder from the Project panel to the Scene tab. In the same way as with the UFO, we will transfer the barn for our cows to the stage.

To make sure that the barn reacts to a collision with a cow when she tries to go inside, you need to add one Box Collider 2D component [13].

Select the shed in the scene and in the Inspector tab, click on the Add Component button. From the list of components, select the Physics 2D section, and in it Box Collider 2D. Make sure there is a check mark next to the "Is Trigger" parameter.

When a cow hits the barn door, we want our trigger to fire. We need to make the collider smaller [14]. Going to the Inspector tab and changing the values of the collider of the Size and Center parameters, we achieve that the collider is located closer to the barn door.

The next step is to add an OnCollision script to make the application react to a collision when the cow enters the barn. This code snippet responds to a collision between a barn and an object named cow (Clone) This is one of the instances of a cow, the prefab of which we will create later. When a collision occurs, a sound is played and the cow object is destroyed.

To play a sound when a cow enters a barn, you first need to attach that sound to the barn. Click on the shed in the Hierarchy tab or Scene, click the Add Component button in the Inspector tab and select Audio Source in the Audio section. We have just added the Audio Source component. Now we need to associate our sound file with it. Click on the circle to the right of the Audio Clip item and select the sound barn. Uncheck the box next to Play on Awake. Drag the cosmic ray image from the Assets tab onto the stage and add a collider to it. This is necessary to detect the collision of the beam with cows. Make sure the "Is Trigger" switch is on.

Next, we'll add the Bullet script [15]. This simple code implies the following. First, we create an AudioClip instance called cowSound, which we will use to store the audio file. This is another option for playing sound if you don't want to bind two audio components to an object. We have declared the variable as public, so we can access it from the Inspector tab. Click on the little dot to the right of cowSound and select the cowSound audio file.

Then we made the ray invisible by turning off its rendering. We use the same object to save resources. This point is very important for optimization for low-power devices.

We detect screen touches, make the beam visible, and reproduce the sound of the beam. When our ray becomes visible, it begins to move downward to hit the cow. We also provided a check to see if our ray was out of the scene. If so, then we put it back in place and our UFO can fire again. The last part checks if the beam has touched the cow. If this happens, a sound is played and the cow disappears. Then the beam becomes invisible again, moves to its original position and the UFO is ready to fire again.

To add a soundtrack to our cosmic ray, you need to select our ray in the Hierarchy or Scene tab and click on the Add Component button in the Inspector tab [16]. Select Audio Source from the Audio section.

Uncheck Play on Awake and click the small dot to the right of the Audio Clip to select a sound file named bizzed.

Drag our cow from the Assets panel to the Scene tab. For the collision to be registered, we must associate the RigidBody2D component with at least one of the objects [17]. It is best to add the component to the cow because the cow can collide with both the barn and the beam. We also need to attach a collider to the cow to detect barn and beam collisions.

Let's create a script to move the cow. The MoveCow class animates the cow as it moves on the screen using the moveSpeed variable. The InvokeRepeating method changes the speed of the cow. This makes the animation more complex. Now that we have attached all the necessary components to the cow, it's time to convert it to a prefab.

A prefab is a reusable GameObject that is stored in the Project View [18]. The prefab can be inserted multiple times in any number of scenes. When you add a prefab to a scene, you create an instance of it. All instances of the prefab are associated with the original prefab and are its clones [19]. No matter how many copies you make for your project, any changes made to the original prefab will be applied to all instances of it.

Let's create a Spawner script that is responsible for displaying cows [20].

To convert a cow to a prefab, drag the cow from the Hierarchy tab to the Assets panel. As a result, the name in the Hierarchy will turn blue.

We use the InvokeRepeating method to clone the cows using the values set for spawnTime and spawnDelay. GameObject cow is set to public and created using the Inspector [21].

To create multiple instances of the cow prefab, use the cow graph that we added to the scene a few minutes ago [22]. Select it and remove its components. Then add the Spawner script.

Figure 4 shows an example of the second computer animation using the Unity game engine.

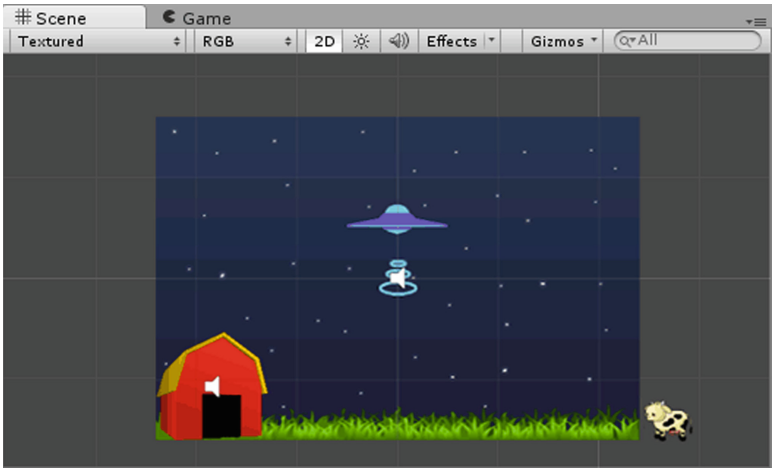

Fig. 4. Example of of the second computer animation using the Unity game engine

7 Conclusion

In this paper, a description of our own method of creating a cartoon in the Unity environment was given as one of the possible ways to develop cartoons, which would give an optimal ratio between the achieved result and the resources used.

There was a need to search for our own solutions, since during a detailed analysis it was revealed that there were no clearly formulated methods for developing cartoons in the Unity environment.

The uniqueness and novelty of the proposed method lies in the synergy of various tools for creating products in the Unity environment that meet the criteria for effective animation. It is worth noting the simplicity and speed of implementation, the clarity and accessibility of the presentation of information, which make it possible to use this method for educational purposes when teaching students of any age to create animation and awaken interest in the production of cartoons.

This method is ideal for studying the technologies of creating animation, for organizing high-quality educational work using animation at the stage of mastering a theoretical and practice-oriented course of academic disciplines.

References

1. Bikmullina, I.: Barkov Instrumentation and Control System for Test Bench. International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon). EEE (2019) https://ieeexplore.ieee.org/author/37087040423

2. Laptev, O.S., Bikmullina, I.I.: Sightseeing Application Based on Location Marking and Convolutional Neural Network Building Recognition. International Russian Automation Conference (RusAutoCon). Conference Paper, Publisher: IEEE (2020) https://ieeexplore.ieee.org/search/searchresult.jsp?contentType=conferences&queryText=RusAutoCon&highlight=true&returnFacets=ALL&returnType=SEARCH&matchPubs=true&refinementName=Author&refinements=Author:Ilsiyar%20Bikmullina&refinements=Author:Ilsiyar%20I.%20Bikmullina
3. Bikmullina, I., Garaeva, E.: The Development of 3D Object Modeling Techniques for Use in the Unity Environmen. International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon) (2020) https://ieeexplore.ieee.org/search/searchresult.jsp?contentType=conferences&queryText=FarEastCon&highlight=true&returnFacets=ALL&returnType=SEARCH&matchPubs=true&refinementName=Author&refinements=Author:I.%20Bikmullina&refinements=Author:Ilsiyar%20Bikmullina
4. Bikmullina, I., Ilnar, V.: Multi-Service Electronic Library of Scientific and Technical Articles. International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon) (2020). https://ieeexplore.ieee.org/search/searchresult.jsp?contentType=conferences&queryText=FarEastCon&highlight=true&returnFacets=ALL&returnType=SEARCH&matchPubs=true&refinementName=Author&refinements=Author:I.%20Bikmullina&refinements=Author:Ilsiyar%20Bikmullina
5. Bikmullina, I., Kusyumov, N.: Orienteering Mobile App. Proceedings of the International Russian Automation Conference, RusAutoConf2020, Sochi, Russia (2020). https://doi.org/10.1007/978-3-030-71119-1_6
6. AYu, D., Kudinov, A.V.: Computer graphics. Tutorial. Tomsk Polytechnic University, Tomsk, p. 160 (2005)
7. Artem, A.: Surface modeling in SolidWorks. https://sapr.ru/article/16361
8. Vasiliev, V.E., Morozov, A.V.: Computer graphics: textbook. SZTU, SPb, p. 101 (2005)
9. Unity engine—features, advantages and disadvantages. https://cubiq.ru/dvizhok-unity/
10. Hawking, J.: Unity in action. Multi-platform development in C, Peter, SPb, p. 336 (2016)
11. Yazev, Y.: Review of the most popular game development engines. https://xakep.ru/2014/09/05/game-development-engines-review/
12. Blender 2.80 Manual. https://docs.blender.org
13. Blender. https://www.blender.org/about/license/
14. Cinema 4D. https://www.maxon.net/ru/produkty/cinema-4d/obzor/
15. GameMaker: Studio. https://la.by/game-engines/game-maker-studio
16. NURBS. http://www.ray-tracing.ru/articles249.html
17. Rendering: Theory. https://render.ru/ru/articles/post/10102
18. Unity Core Platform. https://unity.com/ru/products/core-platform
19. Unity. ProBilder. https://unity3-d.com/ru/unity/features/worldbuilding/probuilder
20. Unreal Engine. https://www.unrealengine.com/en-US/features
21. Vuforia Engine: developer portal. https://developer.vuforia.com/
22. Mytnikov, A.N., Mytnikova, E.A., Kuznetsova, L.N., Solin, S.: Mobile application development technologies. Theory and practice of modern science **4**(10), 504–507 (2016)