УДК 21474

A COMPUTER GAME IN THE SIMULATOR GENRE USING UNITY

Vrakina Karina,
Student
Kharkov national university of radioelectronics
Vrakina Viktoriia,
Associate Professor
National technical university “Kharkov Polytechnic Institute”
Kharkov, Ukraine

Annotation: Nowadays, video games are gaining popularity and development, in particular, the genre of incremental games is very relevant. It is gaining popularity in the mobile gaming market and dominates many flash portals, solving many revenue management tasks, focusing on making decisions that increase revenue.

Keywords: development, mobile game, idle game, clicker, encryption, Unity.

An idle game is a computer game in which the gameplay consists of performing simple actions, such as repeatedly tapping the screen. Clicks are usually performed to earn game currency. In some games, you constantly press on the screen and do not need (provided you buy various improvements), so the currency is earned by itself, including without a player. Hence the name "idle game".

There are many reasons why incremental games have become so popular, and there are many types of different players who play these types of games.

- A sense of accomplishment every time they return to the game.

- Maximum game optimization to the most optimized before closing the game to return to idle mode.

- A clear sense of progress in your game.

- Less physical exposure from players can be more enjoyable for certain people.

- Trophies / badges / achievements form a clear sense of achievement.

- Clear goals in the game, whether a battle with the boss, or a certain level in the game form checkpoints.

- The feeling of growth that arises during the update, always attracts players.

- At the simplest level, people like to watch the numbers increase.

Idle games are a new type of game that is rapidly gaining popularity in the mobile gaming market. This is an unusual genre that no one expected to dominate many flash portals and top mobile charts. Idle games are completely dedicated to revenue flow management. Like simulators, they focus on making decisions that increase revenue.

In many mobile games, the mechanics are built so that the player feels punished when he returns to the game after a break of a few days. For example, in FarmVille plants wither. If you do not return to the game in time, your cultures may perish. All these mechanics are strong enough and give the player good reasons to return to the game, but they also give him a reason to quit the game.

Idle games do not suffer from this. Every time a player returns to a game, he finds huge amounts of money in it. This is perceived as a reward for leaving the game. The player may be absent day, week or month, and the longer this period, the stronger his income. This has created problems in most economies. But not in Idle games. Thanks to the exponential growth curve, the player can leave the game completely painlessly, and his income will continue to grow slowly.

The ideal mobile session should be able to naturally "push" the user out of the game. In the long run, this only stimulates his return. Strong mobile games provide good reasons to return to the game and no less good reasons to leave. This natural "kick from the game" Idle-games have mastered perfectly.

Often the first step in an incremental game is to click or click somewhere in the game, this click generates a certain form of currency that can be used later. Some clickers include an initial tutorial, where others are designed so well that the user immediately starts clicking and buying. This leads to the next purchase feature. A good Idle game should have a way of spending game currency and buying the products that the player wants. These products help the player get more currency per

click or on hold, using the minute or hour indicator. The player then goes through the process of buying more and more products that allow him to earn more currency. This should have a number of items that can be purchased that help generate more currency so that the cycle can be created and continued.

Updates are one of the most important parts of the Idle game, without them the game would be very linear and would not have depth. This can be in the form of an update of the click itself, so each click will result in additional currency, or you can purchase updates for products that make them more effective over time. For example, you may have a pickaxe and you buy upgrades for sharpening, which makes the pickaxe twice as effective.

One of the most interesting things about creating a good incremental game is that it should be fun without playing the game. If the game is set up so that the player must be present for a long time for the game to develop, the game in standby mode is not served in this area. Therefore, it is very important to have a powerful game design when the game is not played. This means that when a player returns, he is rewarded with a large amount of currency to spend and enjoy. The great Idle game will continue to generate game currency until you play, so when you return it will be very useful. Receiving a large amount of currency when you return is a reward in itself, and it looks like an achievement that can also be used to earn future rewards.

Many modern popular mobile games use incremental mechanics. Extremely successful Clash of Clans looks like a military strategy, but the combat mechanics itself is really simple and is a small part of the game. The main part of the game is to improve the village at the expense of gold and elixir, each of which is collected on its own. But you can make them accumulate faster with incremental improvements. Hay Day, made by the same company, is still easier. The main game cycle is exclusively incremental growth and recurring investment of resources.

Cookie Clicker has become one of the most successful modern incremental games. This computer game was created by French programmer Julien Tienno in 2013. Originally it was a browser game, but in 2021 it was released on Steam. The game is also available on Android and iOS platforms. First, the player clicks on a

large cookie icon in the left corner of the screen, earning one cookie for each click. With these cookies, the player can buy new game items. A game that drew public attention to games focused on incremental mechanics. Cookie Clicker is just one currency that a player can slowly accumulate by clicking, or spending on upgrades to increase the speed of automatic liver production. A simple idea, an attractive art style and a gradual immersion into the absurdity of the "plot" - all this made the game, and the whole genre, popular. She has launched a whole wave of similar games that are still coming out and are still trying to bring something new to the concept.

Idle games can even become helpers in real life, offering a plot relevant to humanity in everyday life and life situations, on the example of which the user can learn a useful lesson, preventing mistakes in real life. That is, for example, a more down-to-earth Idle game, where tasks are the ones you could do in real life, and money is not an astronomical amount, but more task-oriented. It's as if someone found a way to start making small money (and then invest it in big money) and made it a game instead of leadership.

Thanks to an engine like Unity, there will be no difficulties in creating an Idle game. Unity is a cross-platform computer game development environment developed by the American company Unity Technologies. Unity allows you to create applications that run on more than 25 different platforms, including personal computers, game consoles, mobile devices, Internet applications and more. Unity was released in 2005 and has been constantly evolving since then. The main advantages of Unity are the presence of a visual development environment, cross-platform support and a modular system of components. Unity has written thousands of games, programs, visualizations of mathematical models that cover many platforms and genres. In this Unity is used by both large developers and independent studios.

The default MonoBehavior class is the base class from which each Unity script is derived. When you create a C # script in the Unity project window, it is automatically inherited from MonoBehavior and provides you with a template script.

Unity's GameObject class is used to represent everything that may exist in a scene. GameObjects are the building blocks for Unity scenes and act as a container

for the functional components that determine what GameObject looks like and what GameObject does.

SHA (Secure Hashing Algorithm) is one of many cryptographic hash functions. Simply put, a hash is a fairly long (64 characters) sequential string of characters and numbers that doesn't matter. And at first it is absolutely normal data from which the special program - hash function creates this unique combination. The peculiarity of hashing is that it completely hides the original data, and the introduction of the same data leads to the same result. A cryptographic hash is like a signature for a data set. If you want to compare two sets of raw data (file source, text or similar), it is always better to hash them and compare the values of SHA256. It's like fingerprints. Even if only one character is changed, the algorithm returns a different hash value. The SHA256 algorithm generates an almost unique fixed size hash of 256 bits (32 bytes). A hash is a so-called one-way function. This makes it suitable for verifying the integrity of your data, hash authentication, protection against unauthorized access, digital signatures, blockchain.

Thanks to the latest hardware improvements, it has become possible to decrypt the SHA256 algorithm back. Therefore, it is no longer recommended to use it for password protection or other similar uses. A few years ago, you protected your passwords from hackers by storing the encrypted SHA256 password in your database. This is no longer the case.

The SHA256 algorithm can still be used to make sure you get the same data as the original. For example, if you download something, you can easily check for data changes due to network errors or malware injections. You can compare the hashes of your file and the original one, which is usually provided on the website from which you receive the data or file.

SHA-256 is one of the following hash functions of SHA-1 and is one of the strongest hash functions available.

To understand the features of hash algorithms, it should be noted that the hash sum is an alternative encryption technology with the inability to reverse decrypt data. This is a one-way combination of ciphers for almost any amount of data. The

technology of creating these algorithms is tied to the basics of the Merkel-Damgard method, within which the data are evenly distributed in groups, each of which goes through a process of information compression. The result of compression is a reduction in data length.

Of the advantages, it should be noted that the SHA-256 algorithm is the most popular and most common mining algorithm in a competitive environment. It constantly demonstrates in practice its extraordinary resistance to hacker attacks, and with its help cryptocurrency users establish effective mining and solve a number of other problems.

In software engineering, a design template is a common recurring solution to a common problem in software development. A design template is not a ready-made design that can be converted directly into code. This is a description or template for solving a problem that can be used in many different situations.

Design templates can speed up the development process by providing proven development paradigms. Effective software design requires consideration of issues that may become apparent only later in the implementation process. Reusing design templates helps prevent subtle problems that can cause serious problems, and improves code readability for programmers and architects familiar with templates.

Among the main, I would like to highlight the following patterns: Singleton, Visitor, Strategy, Command.

Singleton is a generating design pattern that ensures that there is only one instance in a class and provides it with a global access point. Singleton solves two problems at once, violating the principle of single class responsibility. Guarantees the existence of a single instance of the class. This is often useful for accessing a shared resource, such as a database.

Visitor is a behavioral design pattern that allows you to add new operations to a program without changing the classes of objects on which these operations can be performed. Visitor describes a common interface for all types of visitors. It declares a set of methods that differ in the type of input parameter that is required to run the operation for all types of specific elements. In languages that support method

overloads, these methods may have the same name, but the types of their parameters must be different.

Strategy is a behavioral design pattern that defines a family of similar algorithms and places each of them in its own class, after which the algorithms can be interchanged directly during program execution. The context stores a reference to the object of a particular strategy, working with it through a common interface of strategies. The strategy defines an interface common to all variations of the algorithm. The context uses this interface to call the algorithm.

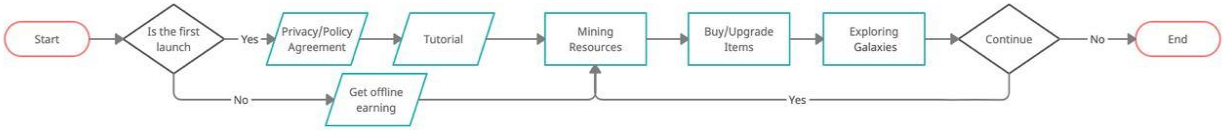
A command is a behavioral design pattern that converts queries into objects, allowing them to be passed as arguments when calling methods, queuing queries, logging them in, and supporting undo operations. The sender saves the link to the command object and contacts it when you need to perform an action. The sender works with commands only through their common interface. He does not know which specific command he is using, as he receives the finished command object from the client. The command describes a common interface for all specific commands. Usually, only one method is described here to run the command.

The purpose of this qualification work is to develop software for mobile devices – Idle games. The user can build entire galaxies, populate them with planets, raise these planets and spread throughout the universe. The user is provided with ten galaxies ready for development and dozens of planets that can be upgraded and developed, in particular, thanks to battery-powered accelerators. The program was written using the Unity engine and C # language. Particular attention was paid to encryption technology, as well as key patterns, including SOLID.

A use case diagram is a dynamic or behavioral diagram in UML. Usage diagrams simulate system functionality with actors and usage options. Usage options are a set of actions, services, and functions that a system must perform. In this context, a "system" is something that is being developed or operated, such as a website. "Actors" are people or organizations that perform certain roles in the system. Usage diagrams are valuable for visualizing the functional requirements of a system that will influence design choices and development priorities. They help to identify

any internal or external factors that may affect the system and should be taken into account.

A use case diagram of the Idle game using Unity is shown on the (pic. 1).



Pic. 1. The use case diagram

The user is greeted by a message on the home screen, then pops up where the user must accept the privacy policy in order to start the game. Next, a window will appear asking you to start the game, where you can click "OK" and continue. The user will see a window, the so-called guide here, with an indication of where to click, namely, the button with the image of the planet to start the exploration. Next, a window with a list of planets will appear, the first users will be available only 1. You need to click "grow a planet" and go to grow a planet. In the field of the main screen of the game, a planet appears, which the user has to click on, thus collecting star dust and growing it. When enough stardust has accumulated - the currency of this game, the user will have the opportunity to improve existing planets and discover new ones. Also, connect a variety of boosters that speed up coin collecting. When all the planets are grown in the first galaxy, the user will be able to open the next galaxy and so on until the end of the game. If the user does not enter the game, or leaves it, simply put, is offline, the study of the planets continues and stellar dust is collected even without the user, which is the main feature of incremental games. And when he comes back into the game, he gets his accumulated coins.