Черкаський національний університет імені Богдана Хмельницького

Кафедра інформаційних технологій

КУРСОВА РОБОТА

З дисципліни "Програмування та алгоритмічні мови"

На тему: "Гра "Башенний кран""

Студента 1 курсу КН-19 групи
спеціальності 122 Комп'ютерні науки
Булавко К.М.
Керівник: ст. викладач Веретельник В.В.

Національна шкала_____

Кількість балів: _____ Оцінка: ECTS____

Члени комісії _____ _____

(підпис) (прізвище та ініціали)

_____ _____

(підпис) (прізвище та ініціали)

_____ _____

(підпис) (прізвище та ініціали)

м. Черкаси – 2020

Зміст

Вступ

В наш час комп’ютерна гра стала найпопулярнішою розвагою в повсякденне життя. Мільйони користувачів проводять більшу частину вільного часу у віртуальному світі. Ігри з’явилися досить не так давно, в другій половинні XX столітті й тоді ж отримали значну популярність серед користувачів, взагалі можна прикинути, що повинні більшу частину заполонити діти, але це не зовсім там, ще значну частину посідають й дорослі, доволі дивно, але так було. Комп’ютерні ігри почали стрімко набирати оберти й з цим покращувалась комп’ютерна графіка, покращували інтерфейс, намагалися вдосконалювати механіку та фізику ігор, завдяки й цьому ігри набирали популярність серед користувачів персональних комп’ютерів. Значною частиною, що вплинуло на стрімке зростання рівня ігрових двигунів, поліпшуванні пристрої комп’ютерних інструментів й ігрових рушіїв, за допомогою яких й поліпшувалась значна трата часу й швидше розроблялись відеоігри для користувачів. Сьогоденність значна кількість розробників ігор, що розробляють відеоігри, використовують ігрові двигуни Source, Unity, Unreal Engine, CryEngine, Construct та інших, усі перелічити важко, але кожний з них зменшує витрату часу розробників. Програми при створенні двигуні, зайнялась зрозумілим користувацьким інтерфейсом, удосконаленою симуляцією фізики, але для входження в цю сферу потребує рівень знань, тобто поріг входження високий й не кожен може. Сучасні рушії настільки розвинуті, око людини важко помітити то є насправді чи то вигадка дизайнерів та розробників. Але це впливає комп'ютерне залізо якого недостатньої потужності. Завдяки чому й розробники заліза частіше випускають нові компоненти які виконують з легкість потрібні забаганки програмістів.

Зараз відеоігри поділяються на багато жанрів на будь-який смак тим самим з легкістю задовольняють забаганки користувачів. Десятки тисяч ентузіастів створюють кожного дня фантастичні й неймовірні проекти, завдяки цьому й відеоігри не мов затягують користувача в цей безмежний світ.

Розділ 1. Постановка задачі.

Мета роботи – розробка відеогри Башенний кран, яка симулює дійсну гру.

Задачі роботи:

1. Провести аналіз відео двигунів для виконання поставленої задачі
2. Зробити візуальне середовище в грі
3. Скласти структурну схему гри
4. Скласти структуру схему програмного процесу
5. Реалізувати у вигляді комп'ютерної гри
6. Проаналізувати розроблений продукт

1.1 Середовище для створення гри

Завдання полягало розробити відеогру, а це краще розробляти за допомогою різного софта тобто ігрового двигуна на якому стоятиме весь ігровий продукт. Вибрати саме який двигун не важно, найкращий родич мови C# це Unity який й використовував для розробки. Взагалі можливо використовувати такі двигуни як CryEngine, Unreal Engine. Якщо не вдаватися у дрібниці, в усіх софтах майже все різне, починаючи від фізики та задачі та закінчуючи інтерфейсом та зручності виконанням завдання. Всі софти потребують знання певних мов програмування без цього нікуди. Якщо казати за CryEngine, це також родич C# або C++, Unreal Engine - двигун взагалі використовують для масштабних проектів та великогабаритних ігор. Unity платформа має великі масштаби навчальних матеріалів до яких доступ безкоштовний. Завдяки цьому розробники без проблем мають можливість перенести проект на всі найпопулярніші платформи та операційні системи. Unity дає змогу створювати власні ігри завдяки повноцінній і гнучкій платформі розробки в реальному часі. Unity дозволяє швидко створювати, підтримувати і монетизувати гру всіма можливими методами.

Отже, для початку написання гри вирішено вибрати двигун Unity та ще він крос-платформний та й він підходить завдяки 2D гри, ніж інші рушії.

1.2 Огляд правил гри “Башенний кран”

На екрані зображений башенний кран яким користувач повинен побудувати споруду. Кран має тра рухливі складові.

- Кран стоїть на рейках за допомогою і чого може пересуватися вліво і право.
- Візок найголовніша частина крана за допомогою якої і переміщається вантаж по стрілі крана.
- Гак - це складова візка за допомогою чого і можна переміщати предмети, їм зачіпається та відчіпляєтеся предмет, піднімається і пересувається на потрібне місце.

На галявині знаходяться кілька предметів. Для полегшення гри, розробник заздалегідь підготував місцевість для будинку і позначив візуально на вигляд, це бетон, далі це три стінки і найголовніший об'єкт без чого не може бути будинок, це дах у формі рівнобічної трапеції.

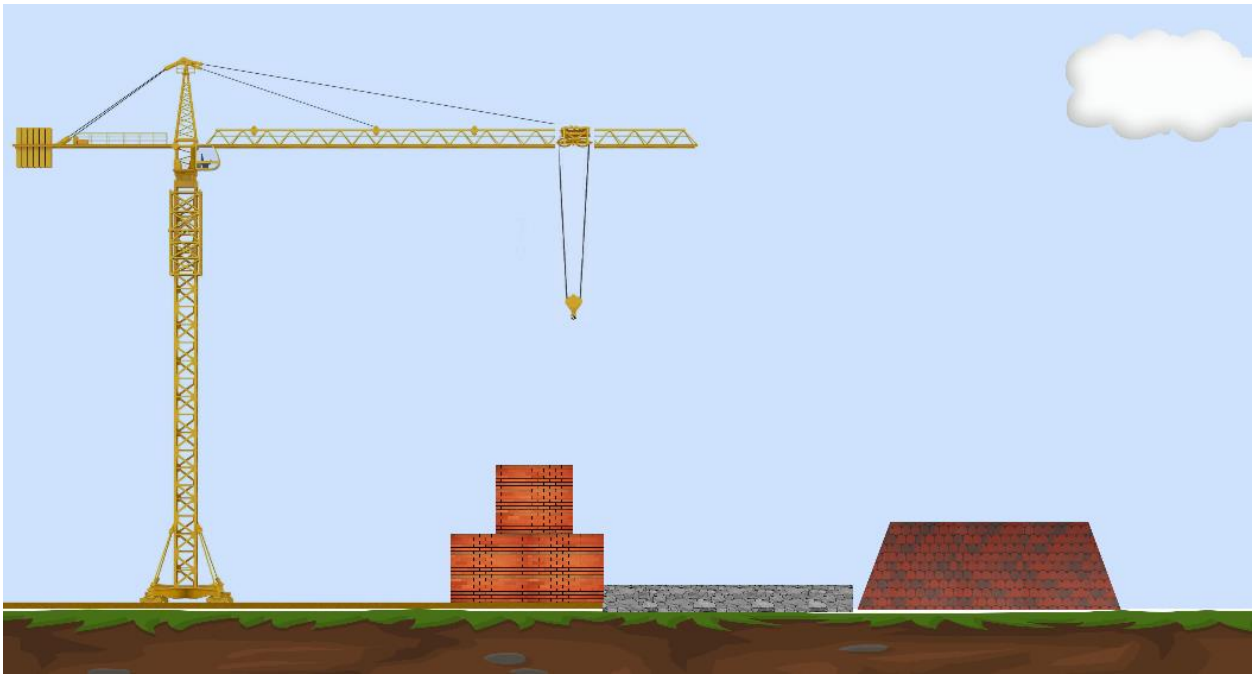

Рис. 1.1 Приклад розташування при початку гри

Отже, для відеогри та для початку, цих об'єктів вистачить для побудови будинку.

1.3 Висновки

Отже, головна мета та задача роботи – створення відеогри “Башенний кран” за допомогою Unity в 2D форматі й крос-платорменного рушія, правила гри, проектування, алгоритм ігрового процесу та реалізація.

Розділ 2. Проектування гри.

2.1 Проектування загальних елементів

Мета розробки програми – створення гри в якій Башенний кран повинен побудувати будівлю за принципом гри “Тетріс”, то потрібно написати наступні елементи.

- Переміщення об’єктів вліво та вправо
- Підняття та відпускання об’єктів
- Реалізувати механізм аби кран не завалився
- Та ще додатково реалізую, цього не будо в завданні
- Головне меню
- Кнопки “Play” та “EXIT” для зручності користувачу

2.2 Проектування меню

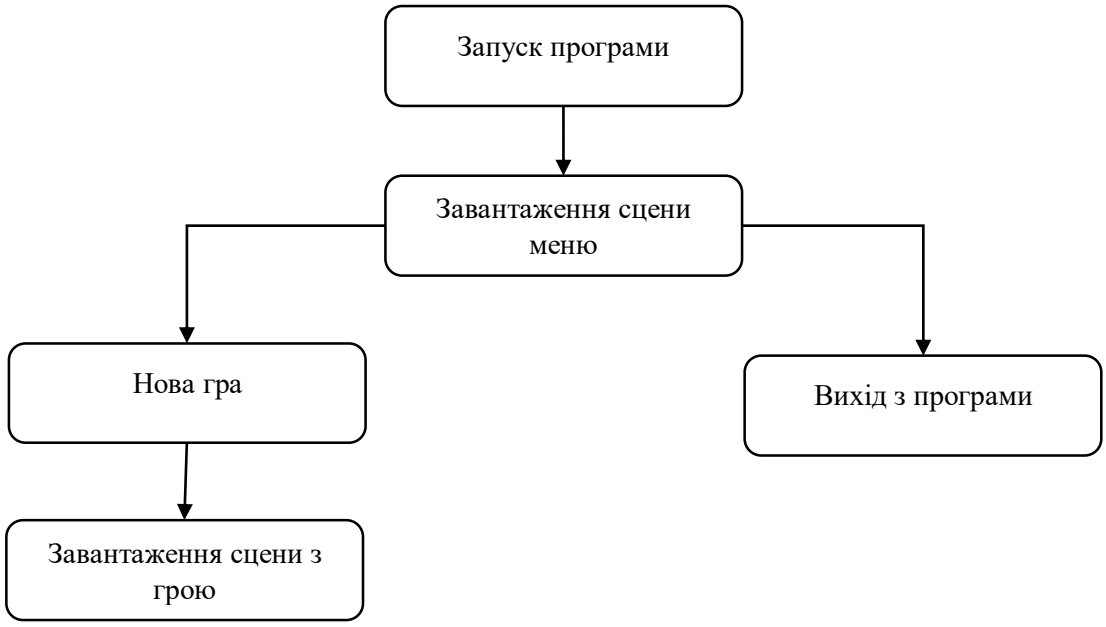

Рис. 2.1.Схема взаємодії з головним меню

2.3 Проектування алгоритму ігрового процесу

Схема ігрового процесу буде реалізовано таким чином:

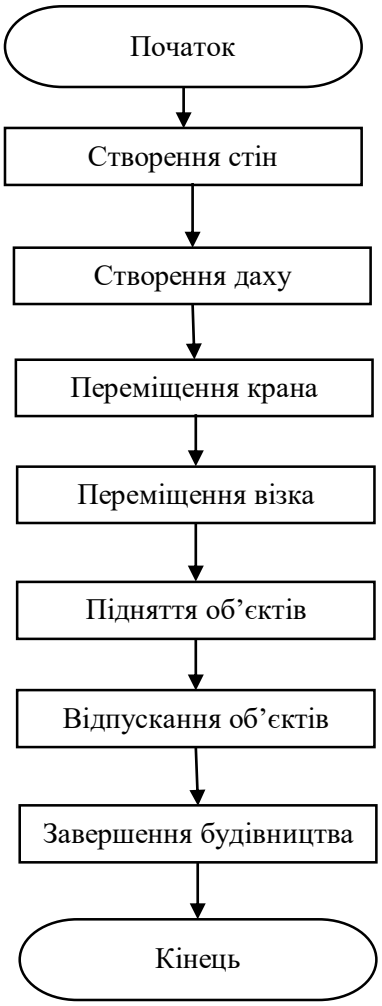

Рис. 2.2 Загальна блок-схема ігрового процесу

2.4 Висновки

Отже, в цьому розділі спроектовано важливі основні моменти програми, які за допомогою Unity середовища реалізуються. Розробили простий блок-схему процесу гри за взаємодія с об’єктами та ігровим меню.

Розділ 3. Реалізація і тестування програми.

За допомогою схема раніше спроектованих реалізуємо програмну гру, опишемо функції та методи, та напишемо головне меню для користувача.

3.1 Процес розробки ігрового меню

Для початку ми створимо проект Unity в якому і почнемо розробку. С початку створюємо саму сцену, додаємо елемент Canvas, в якому вже додаємо колір тла або зображення image саме це й стане основою головного меню.

Щоб трошки прикрасити однотонність фону можемо додати кілька цікавих картинок (Sprite) для цього потрібно просто перемістити ці спрайти по верх нашого тла.

Найголовнішою частиною для користувача є дві кнопки PLAY та EXIT, на них потрібно прикріпити скріпт main_menu з функціями NewGame() і QuitGame() логічно.

Рис. 3.1. Головне меню програми

Лістинг скріпта має наступний вигляд:

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

public class Main_Menu : MonoBehaviour
{

    public void NewGame()
    {
      SceneManager.LoadScene("Game");
    }

    public void QuitGame()
    {
       Application.Quit();
    }
}
```

Метод SceneManager.LoadScene() – він відповідну сцену Game завантажує.

Application.Quit() – Логічно, завершує дію програмного продукту (відеогри).

3.2 Процес розробки інтерфейса під час гри

Створимо нову сцену яку назвимо Game. Також додаємо в неї об’єкт Canvas з компонентом image тобто картинка. Щоб покращити якість інтерфейса гри, Unity подбало про користувачів і випустили покращений текст TextMeshPro, вибираємо чітки шрифт чіткого зображення.

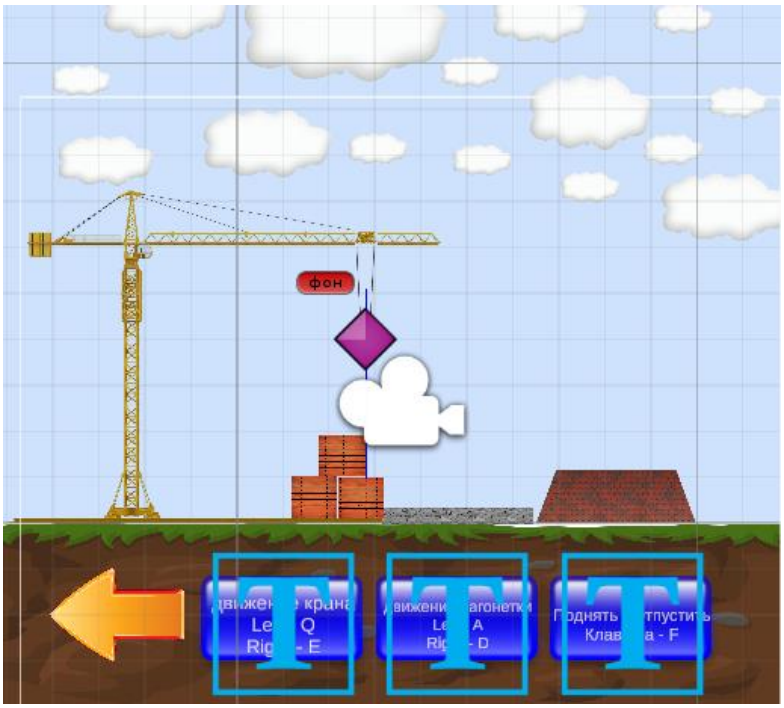

Рис. 3.2. Вигляд інтерфейсу на сцені

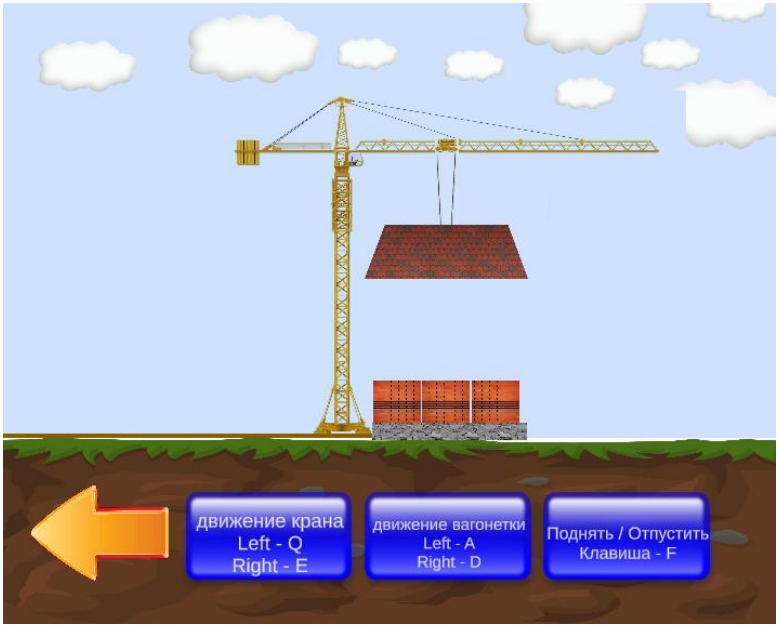

Рис. 3.3. Вигляд інтерфейсу на під час ігрового процесу

3.3 Опис методів і функцій

В цій програмі не застосовується виклик різних скриптів по вимогі, тому описувати почну кожен скрипт окремо. Почнемо з основи. Створимо клас move_crane в якому здійснюється рух крана. Підключаємо using UnityEngine – Юніті двигун завдяки якому й буде здійснюватися. Rigibody2D – це фізика гри.

У Update() – ми вписуємо за допомогою яких клавіш здійснюється рух вліво та право. Задаємо в публічну змінну public float speed = 10f шквидкість з якою буде рухатись. Та щоб це все плавно покадрово виглядало, ми швидкість множимо на Time.deltaTime в результаті чого отримуємо плавне та чітке зображення. Див рис 3.4.

```
using UnityEngine;

public class move_cart : MonoBehaviour
{
    // Start is called before the first frame update
    public float speed = 10f;
    private Rigidbody2D rb;
    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
    }

    // Update is called once per frame
    void Update()
    {
        float moveX = Input.GetAxis("Horizontal");
        rb.MovePosition(rb.position + Vector2.right * moveX * speed * Time.deltaTime);
    }
}
```

Рис. 3.4. Лістинг функції move_crane

move_cart – клас який відповідає за рух візка вліво та право. Як завжди підключаємо бібліотеку using UnityEngine.

Задаємо в публічну змінну public float speed = 5f шквидкість з якою буде рухатись. Тобто тут точно такий же скрипт як й попередній. Але є один нюанс “Vertical” це відповідає за кнопки по горизонту, заходимо в Edit/Project Settings/Input Manager та в пункті Vertical шукаємо Alt Negative Button виставляємо – Q,Alt Positive Button –E й закриваємо вікно. Див рис 3.5.

```
using UnityEngine;

public class move_crane : MonoBehaviour
{
    // Start is called before the first frame update
    public float speed = 5f;
    private Rigidbody2D rb;

    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
    }

    // Update is called once per frame
    void Update()
    {
        float moveX = Input.GetAxis("Vertical");
        rb.MovePosition(rb.position + Vector2.right * moveX * speed * Time.deltaTime);

    }
}
```

Рис. 3.5. Лістинг функції move_crane

Клас falling_objects виконює взаємодію між користувачем та клюком крана тобто підіймає та відпускає об’єкти. bool hold – відповідає чи держить зараз щось крюк. distance – це уявна лінія за допомогою якої ми дізнаємось чи можемо дістати до чогось. RaycastHit2D – використовується для виявлення об'єктів, які лежать уздовж шляху променя. Transform holdPoint – це те місце куди прикріпиться об'єкт. Див рис 3.6.1

```
using UnityEngine;

public class falling_objects : MonoBehaviour
{
    public bool hold;
    public float distance = 10f;
    RaycastHit2D hit;
    public Transform holdPoint;
    //public float throwObject;

    void Start()
    {

    }
```

Рис. 3.6.1 Лістинг функції falling_object

Update() – Функція, що викликається кожен кадр(фрейм). Тут реалізовано, щоб по натиску на кнопку – F об'єкт зачіплявся за гак та підіймався. Юніті дає змогу по фреймам перевіряти потрібну інформацію також тут реалізовано такий баг як уникнення підняття інших об'єктів які не потрібні. Як раз цим й займається if(hit.collidet != null && hit.collider.tag == “stenka_krusha”) Див рис 3.6.2

```
void Update()
{
    if (Input.GetKeyDown(KeyCode.F))
    {
        if (!hold)
        {
            Physics2D.queriesStartInColliders = false;
            hit = Physics2D.Raycast(transform.position, Vector2.down * transform.localScale.y, distance);

            if (hit.collider != null && hit.collider.tag == "stenka_krusha") // к примеру
            {
                hold = true;
            }
        }
        else
        {
            hold = false;

            if (hit.collider.gameObject.GetComponent<Rigidbody2D>() != null)
            {
                hit.collider.gameObject.GetComponent<Rigidbody2D>().velocity = new Vector2(transform.localScale.y, 1);
            }
        }
    }
```

Рис. 3.6.2 Лістинг функції falling_object

Цей If (hold) відповідає за другий натиск на клавішу – F щоб можно було відпустити об'єкт. В юніті колайдери та холд поєнти відіграють важливу роль. Див рис 3.6.3

```
    }

    if (hold)
    {
        hit.collider.gameObject.transform.position = holdPoint.position;
    }
}
```

Рис. 3.6.3 Лістинг функції falling_object

OnDrawGizmos – тут виконується роль відпускання об'єкт з гака. Задається траєкторія й об'єкт по ній падає. Див рис 3.6.4

```
private void OnDrawGizmos()
{
    Gizmos.color = Color.blue;
    Gizmos.DrawLine(transform.position, transform.position + Vector3.down * transform.localScale.y * distance);
}
```

Рис. 3.6.4 Лістинг функції falling_object

3.4 Тестування

Меню в самій грі працює та виконує потрібні функції. Див рис. 3.10.

а

Рис. 3.10. Ігрове меню під час ігрового процесу

Тестуваннями займались декілька користувачів ,в усіх програма коректно працює, з легкістю кран та вагонетка пересовується Див рис. 3.11 та рис. 3.12.

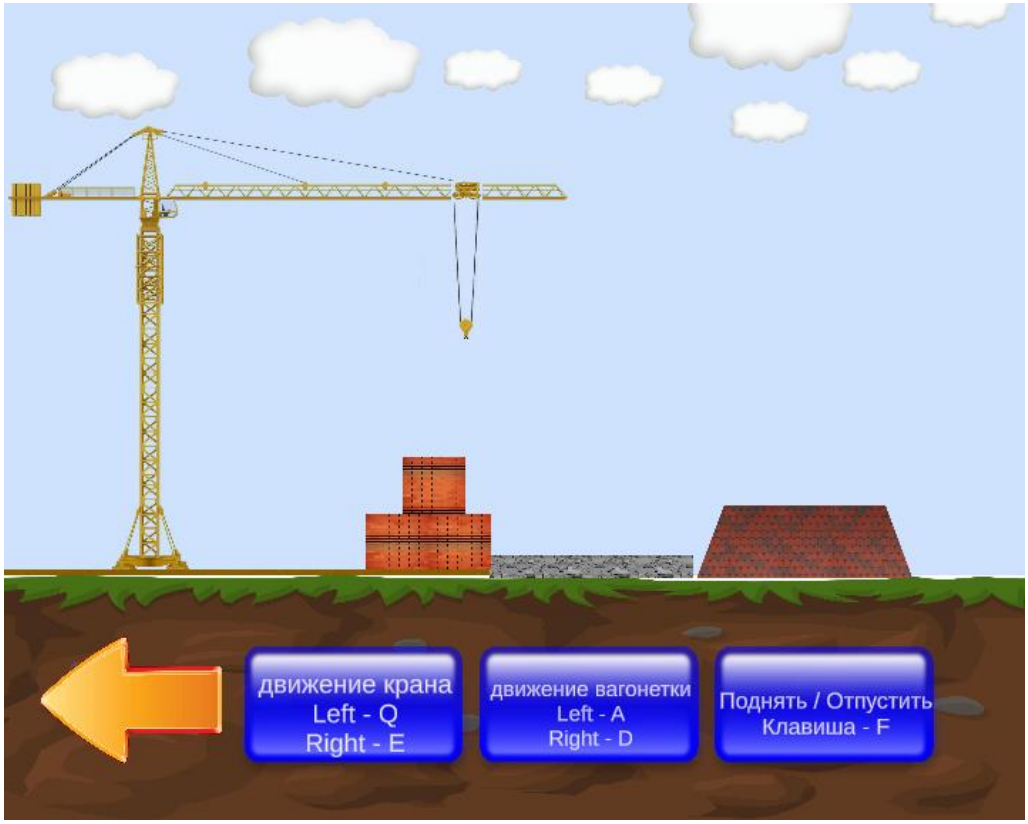

Рис. 3.11. Початкове положення споруди

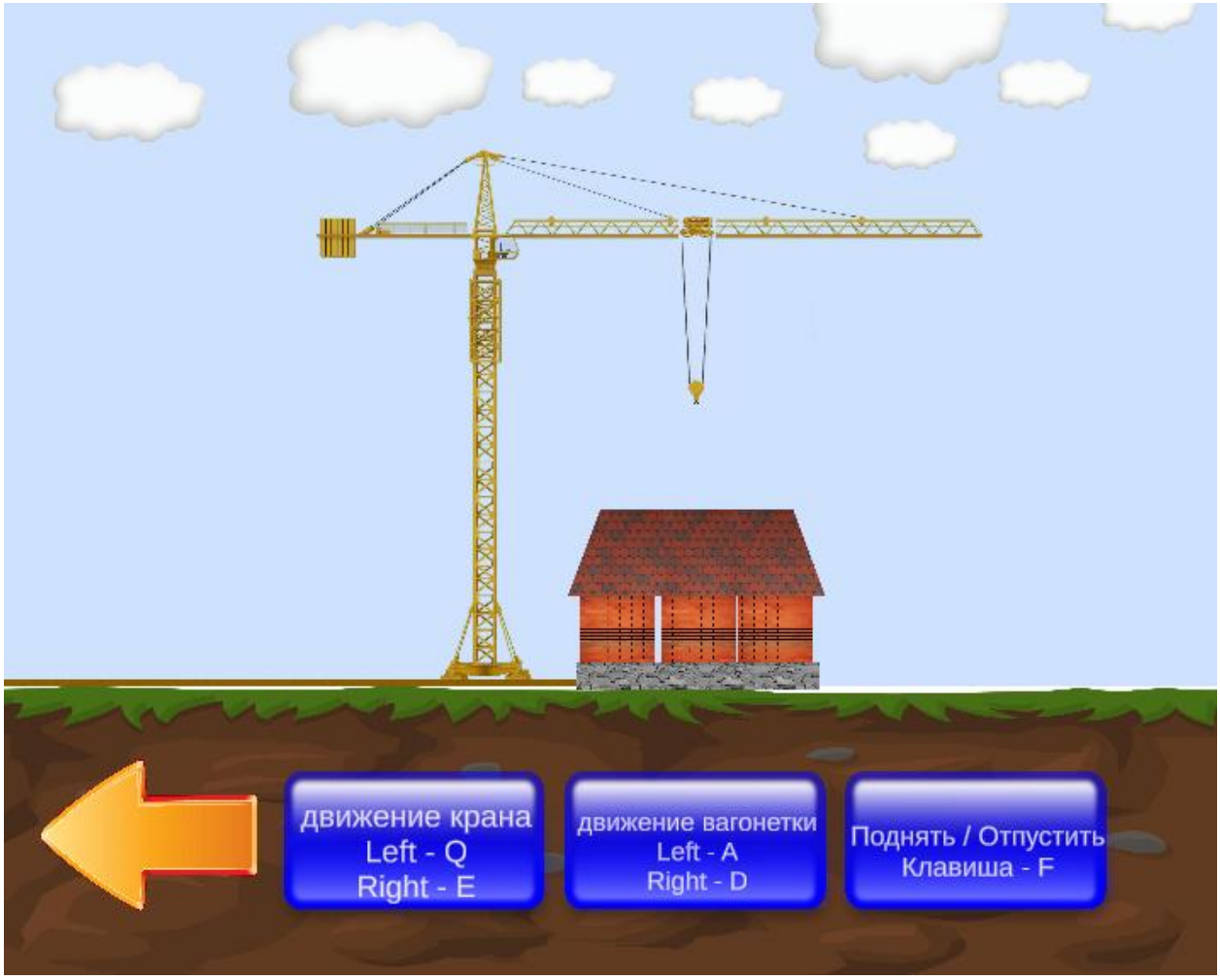

Рис. 3.12. Побудована споруда

3.5 Висновки

Отже, ігрове меню реалізоване та працює коректно, внизу написано яка кнопка відповідає руху. Створили скрипти в яких реалізували потрібні функції та методи. Програмний продукт пройшов у декількох користувачів тестування на знаходження багів.

Висновок

За час розробки було створено програмний продукт або відеогра “Башенний кран”. Гра поки що має простий інтерфейс й зручне просте меню.

Гра в цілом має перспективу і можно реалізувати

- Більш зручніший інтерфейс
- Змінити графіку текстур
- Додати інтерактивний текст
- Музика
- Гайд як користуватись
- Кілька рівнів складності
- Обмеження часу в грі за яке потрібно побудувати будинок

По недолікам можно виділити найголовніший, є тільки один рівень, хотілось би ще додати щоб користувач ввійшов в смак. Також є кілька багів з механікою та фізикою руху крана, але це змінити можливо.

Список використаних джерел

1. Методичні вказівки до виконання та оформлення курсової роботи з дисциплін “Основи програмування”, “Програмування та алгоритмічні мови”, “Алгоритмізація та програмування”. Черкаси: Вид. від ЧНУ імені Богдана Хмельницького, 2016.-32с.
2. Документаційний документ по C# [Електронний ресурс] Режим доступу: https://docs.microsoft.com/ru-ru/dotnet/csharp/ Перевірено 23.05.2020
3. Ще більша документація по C# [Електронний ресурс] Режим доступу: https://metanit.com/sharp/ Перевірено 23.05.2020
4. Документація по Unity [Електронний ресурс] Режим доступу: https://docs.unity3d.com/Manual/index.html Перевірено 24.05.2020
5. Форум з цікавими матеріалами по мові C# [Електронний ресурс] Режим доступу: https://habr.com/ru/search/ Перевірено 23.05.2020
6. Відеоматеріали по Unity2D та C# [Електронний ресурс] Режим доступу: https://www.youtube.com/ Перевірено 24.05.2020