# Лабораторна робота №3.

**Хід виконання роботи:**

До роботи взята Polycode IDE , який використовується для розробки кросплатформних додатків на C++ або Lua. Ядро Polycode написано на C++ і наразі працює на Mac, Windows і Linux з підтримкою iOS і Android. Уся функціональність, що залежить від платформи, абстрагується до кросплатформового ядра ​​і реалізується на кожній платформі, тому той самий код C++ компілюється на кожній підтримуваній платформі з коробки.

Polycode також можна використовувати як суто сценарний фреймворк Lua через Polycode IDE. Програми, розроблені на Lua, можна експортувати безпосередньо до всіх підтримуваних платформ. Polycode є повністю безкоштовним і відкритим вихідним кодом за ліцензією MIT.

Після завантаження з офіційного сайту <http://polycode.org/> починається робота з запуску фреймворку та створення нового проекту Game\_2021 в категорії 2D.

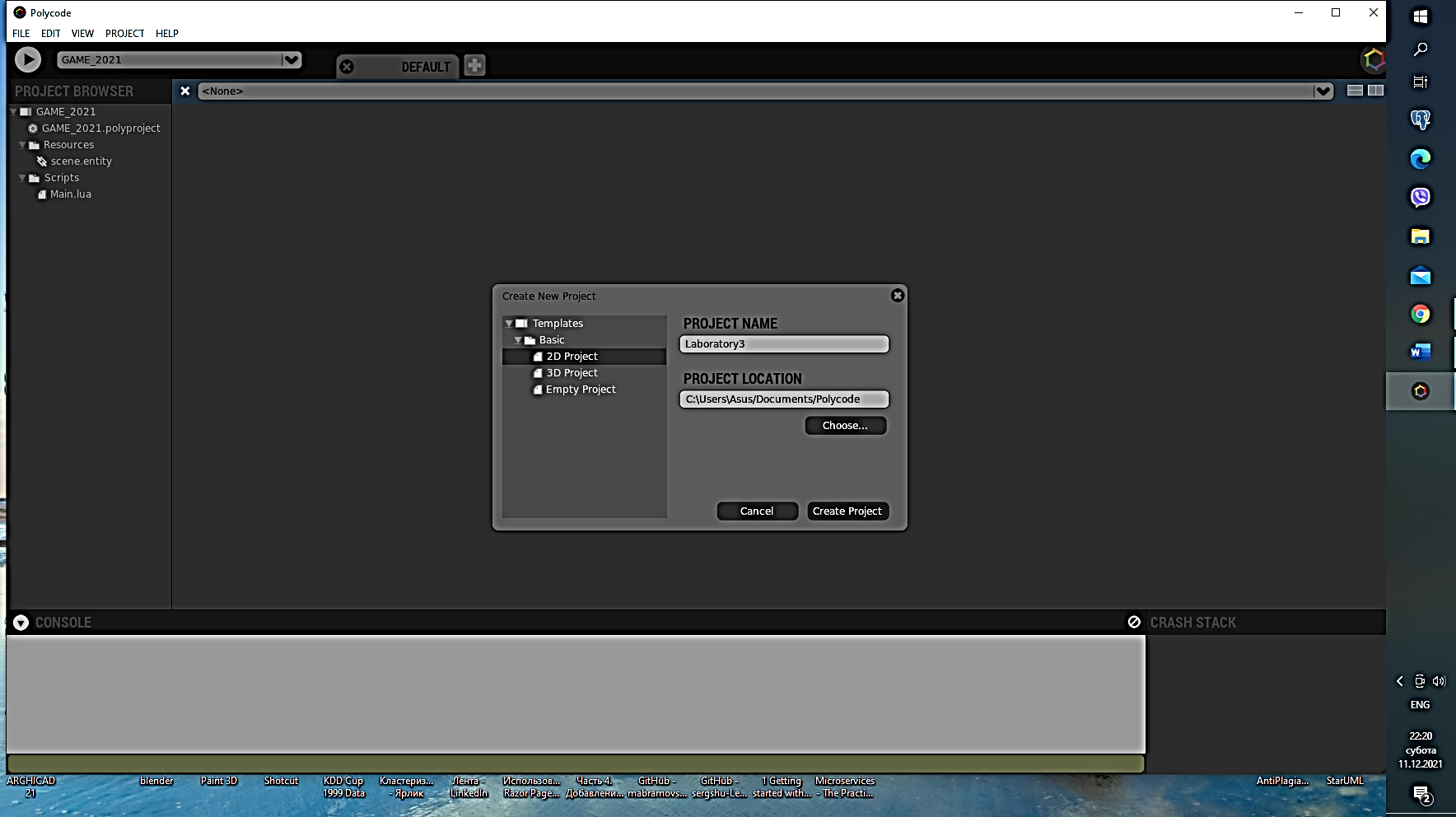


Рис. 3.1. Скріншот створення нового проекту Game\_2021.

Далі відкривається проект, а далі послідовно папка Scripts, де до файлу Main.lua записується такий код:

*scene = Scene(Scene.SCENE\_2D)*

*sun = ScenePrimitive(ScenePrimitive.TYPE\_CIRCLE, 0.2,0.2, 30)*

*sun:setColor(0.9, 0.8, 0, 1)*

*sun.colorAffectsChildren = false*

*scene:addChild(sun)*

*planet = ScenePrimitive(ScenePrimitive.TYPE\_CIRCLE, 0.1,0.1, 30)*

*planet:setPosition(0.3,0)*

*planet:setColor(0.2, 0.8, 0, 1)*

*planet.colorAffectsChildren = false*

*sun:addChild(planet)*

*moon = ScenePrimitive(ScenePrimitive.TYPE\_CIRCLE, 0.05, 0.05, 30)*

*moon:setPosition(0.1,0)*

*moon:setColor(1, 1, 0.6, 1)*

*planet:addChild(moon)*

*planetRotation = 0*

*moonRotation = 0*

*function Update(elapsed)*

*planetRotation = planetRotation + elapsed*

*moonRotation = moonRotation + (elapsed \* 6)*

*planet:setPosition(cos(planetRotation)\*0.3, sin(planetRotation)\*0.3)*

*moon:setPosition(cos(moonRotation)\*0.1, sin(moonRotation)\*0.1)*

*end*

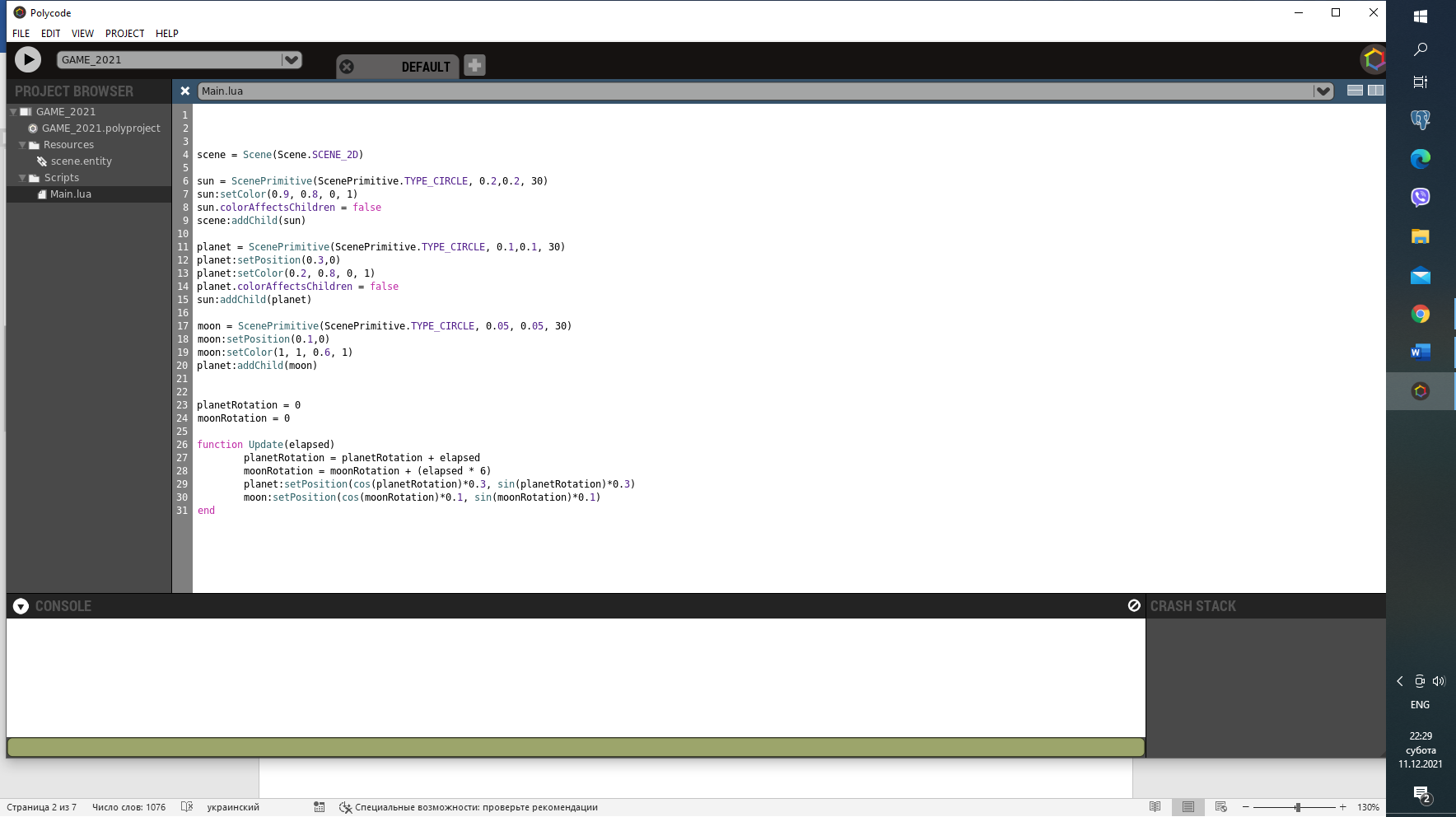
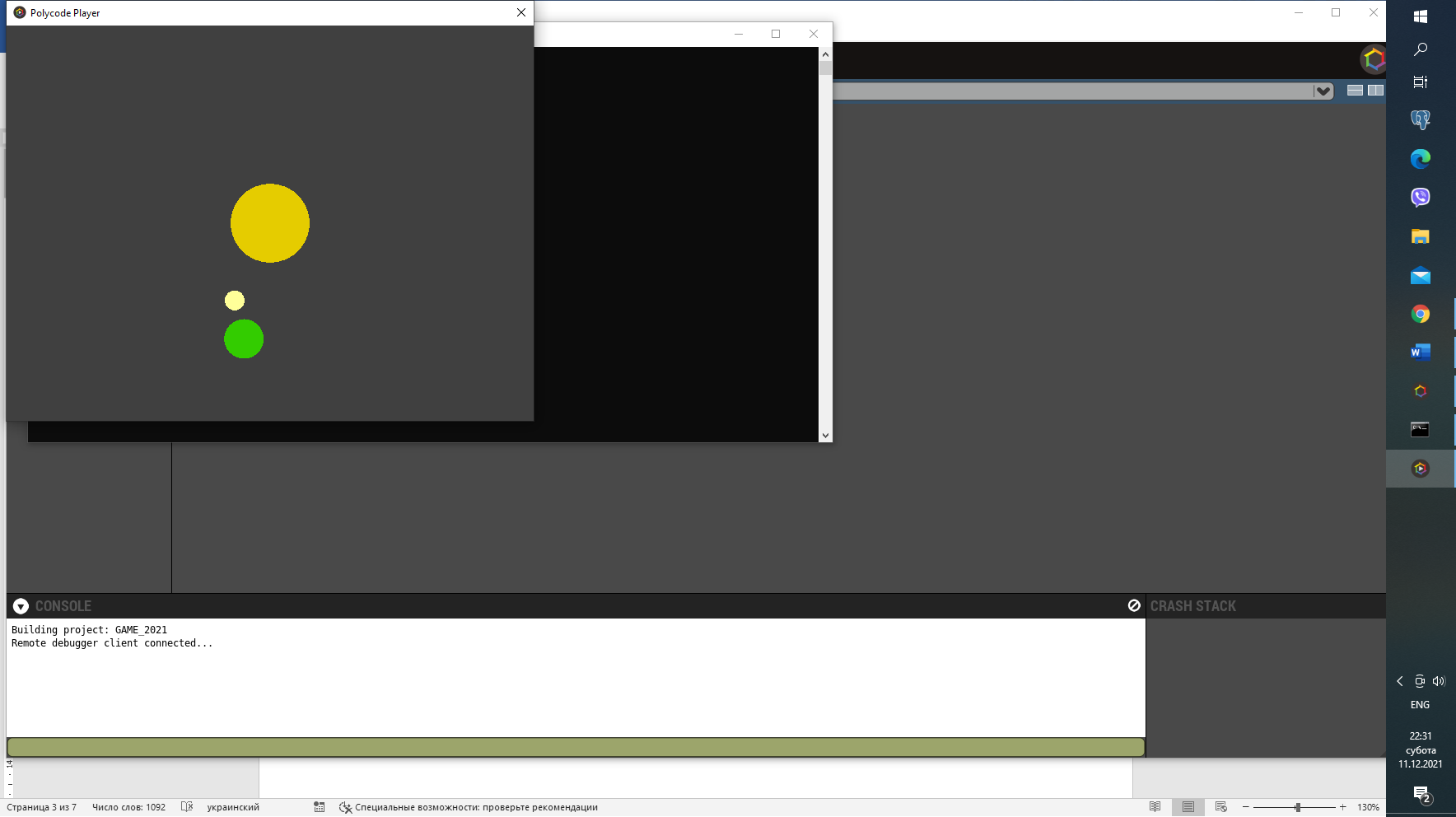


Рис. 3.2. Скріншот файлу Main.lua в програмі Polycode IDE.

Файл Main.lua зберігається, потім сам проект запускається як новостворена гра. Як результат запуску гри, з’являється анімаційна картинка з обертанням планет навколо сонця.



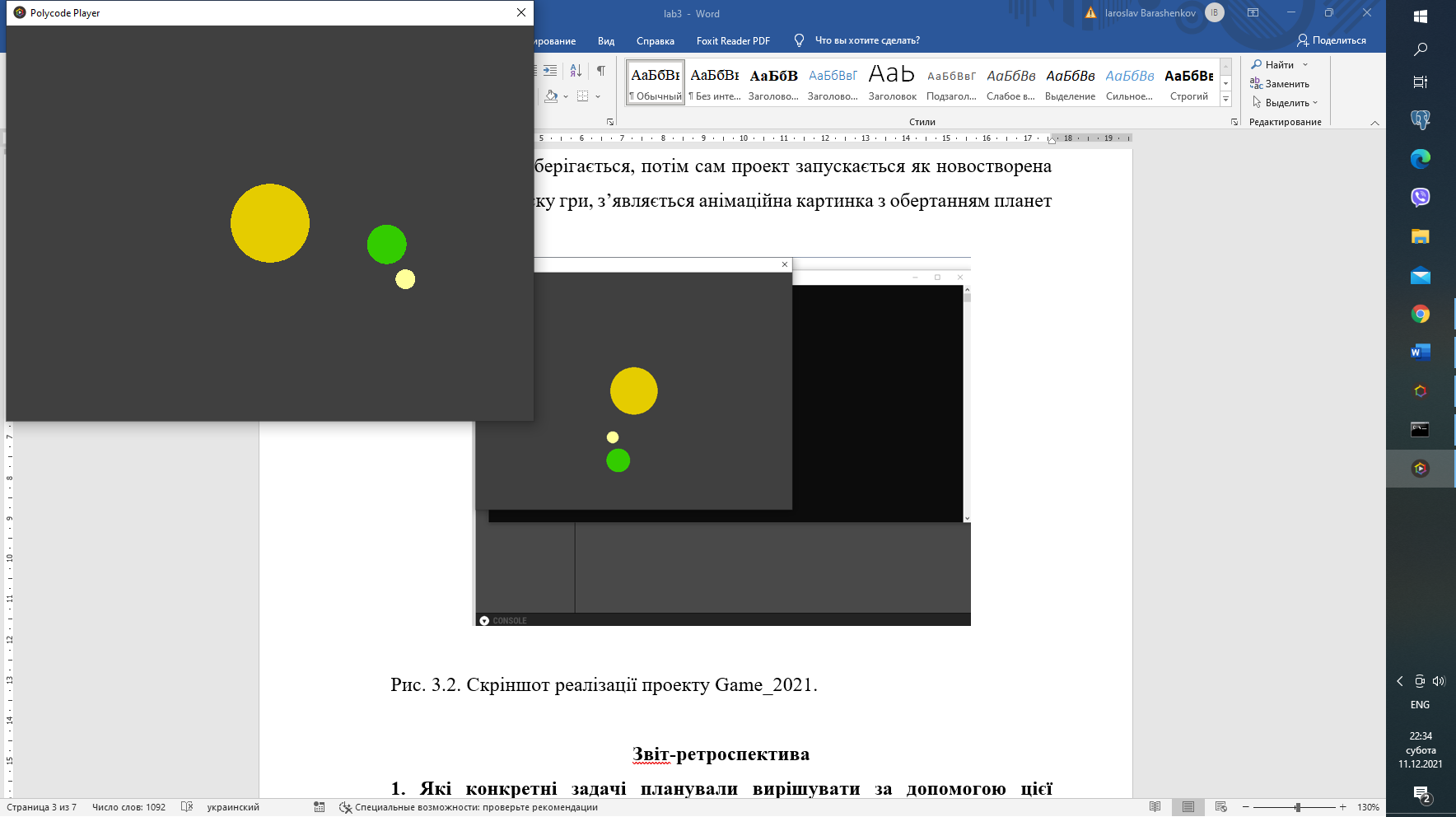


Рис. 3.2. Скріншот реалізації проекту Game\_2021.

# Звіт-ретроспектива

**1. Які конкретні задачі планували вирішувати за допомогою цієї бібліотеки?**

В нашому випадку Polycode IDE дозволяє практично вирішувати завдання по створенню невеликих комп’ютерних ігор, які допоможуть оригінально висвітлювати учбові теми для студентів чи школярів.

**2. Чому було обрано саме цю бібліотеку, а не аналоги?**

Аналогами Polycode IDE є Unreal Engine та Cocos Creator. Перший програмний продукт має надвисокі вимоги до якості відеокарти, кількості оперативної (8 Гб та вище) та загальної пам’яті (44 Гб). Другий програмний продукт не дає можливості оперативно зареєструватися, спроби реєстрації без особливого успіху можуть тривати день.

**3. Наскільки просто та зрозуміло було отримати, встановити, налаштувати та почати використовувати цю бібліотеку?**

Встановлення фреймворку відбувається безпосередньо з головного сайту <http://polycode.org/> та не має ніяких особливих затримок та несподіваних фінансових вимог.

**4. Наскільки зрозумілою та корисною була документація бібліотеки?**

Документація бібліотеки має окремий учбовий розділ, де послідовно описані все етапи навчальної роботи з бібліотекою.

**5. Наскільки було зрозуміло, як саме використовувати бібліотеку, які класи/методи/функції використовувати для вирішення поставлених задач?**

Програма Polycode зосереджена навколо класу Core, який ініціалізується екземпляром PolycodeView .

Ядро - це абстрактний клас, який визначає пов’язані з платформою функціональні можливості, такі як оновлення, керування вікнами, операції з диском, керування потоками та введення, і реалізується як підклас на кожній підтримуваній платформі. Щоб ініціалізувати програму Polycode, треба створити екземпляр Core, передавши йому деякі початкові параметри та вказівник на екземпляр PolycodeView.

PolycodeView - це клас, який містить інформацію про вікно, специфічну для платформи. Він реалізований абсолютно по-різному на кожній платформі, щоб відповідати парадигмі програмування цієї платформи.

**6. Наскільки зручно було використовувати бібліотеку, чи не треба було писати багато надлишкового коду?**

Щоб створити нову програму Lua в IDE, виберіть Файл→Новий проект у меню програми та виберіть шаблон для початку.

Не потрібно писати код для ініціалізації програми Lua, вона працюватиме навіть із порожнім вихідним файлом, оновлюється й відтворюється автоматично. Щоб запустити програму, натисніть велику кнопку відтворення або виберіть « Проект»→ «Запустити проект» у меню програми.

**7. Наскільки зрозумілою була поведінка класів/методів/функцій з бібліотеки?**

Цикл застосування в C++. Після створення екземпляра Core його необхідно оновити в циклі. На більшості платформ, просто в той час як цикл буде досить, хоча на деяких платформах, як Mac, цикл виконується з допомогою таймера какао. Перегляньте шаблон проекту для кожної платформи, щоб дізнатися більше про виконання циклу.

Найпростіший спосіб оновити Core - це викликати updateAndRender на ньому. Це оновить усі внутрішні системи Polycode, а потім відобразить активні сцени на екрані. Цей метод поверне внутрішній запущений стан Polycode, і

updateAndRender — це в основному зручна обгортка, яка викликає Update() і Render() одним методом, але в деяких випадках вам потрібно розділити виклики Update і Render і запустити деякий код після оновлення, але до того, як все буде відтворено.

Оновлення, у свою чергу, є зручною оболонкою для внутрішньої фіксованої системи оновлення Polycode. Якщо програма використовує фізичний модуль або виконує будь-яке детальне виявлення зіткнень, радимо використовувати фіксований цикл оновлення у своїй програмі разом із фіксованим оновленням Polycode. Для цього потрібно спочатку викликати systemUpdate() на ядрі, а потім продовжувати викликати fixedUpdate, доки він не поверне false, виконуючи між ними ваш код. Polycode гарантує, що це відбуватиметься лише через фіксований інтервал, визначений частотою кадрів, на яку було встановлено ядро. Цей цикл також дозволяє реагувати на фізичні події, які відбуваються всередині Polycode під час фіксованих оновлень.

**8. Наскільки зрозумілою була взаємодія між різними класами/методами/функціями цієї бібліотеки, а також взаємодія між бібліотекою та власним кодом?**

Polycode використовує керовану систему графіків сцени для всього свого внутрішнього візуалізації, що означає, що ви не відтворюєте речі вручну, а додаєте їх до сцени, і візуалізація виконується автоматично. Сцени та сутності є базовими компонентами цього графіка сцени.

Вся візуалізація в Polycode відбувається через сцени. Сцена, по суті, є контейнером для кореневої сутності та камерою, яка визначає спосіб відтворення сцени. Сцени можна створити як 2D або 3D, що визначає початкові налаштування камери за замовчуванням для сцени (ви можете змінити налаштування камери в будь-який час після цього або відтворити за допомогою інших камер). Ви можете одночасно відтворювати будь-яку кількість сцен. Наприклад, ви можете використовувати 3D-сцену для візуалізації гри і 2D-сцену поверх неї для відтворення інтерфейсу користувача. Сцени керуються внутрішньо за допомогою SceneManager.

Сутності є основою всіх позиційних і відтворюваних об’єктів у Polycode. Вони містять інформацію про перетворення, візуалізацію та оновлення коду, а також різні властивості сутності, такі як ідентифікатори та теги. Вони можуть містити інші сутності та утворювати складні ієрархії. У кожній сцені міститься один корінь, і всі сутності, додані до цієї сцени, є дочірніми її кореневими сутностями.

**9. Чи виникали якісь проблеми з використанням бібліотеки? Чи вдалось їх вирішити, як саме?**

Проблем не виникало.

**10. Що хорошого можна сказати про цю бібліотеку, які були позитивні аспекти використання бібліотеки?**

Бібліотека не потребує вживання великих мережевих та апаратних ресурсів.

**11. Що поганого можна сказати про цю бібліотеку, які були негативні аспекти використання бібліотеки?**

Бібліотека дозволяє створювати лише дуже примітивні за дизайном ігри

**12. Якби довелось вирішувати аналогічну задачу, але вже враховуючи досвід використання в цій лабораторній роботі, що варто було б робити так само, а що змінити? Можливо, використати інші бібліотеки, чи використати інші можливості цієї бібліотеки, чи інакше організувати код, чи ще щось?**

Цю бібліотеку неможливо використовувати в якомусь іншому програмно-поведінковому аспекті, тому хід виконання розробки програми ігри важко змінити.

Джерела:

1. <https://github.com/ivansafrin/Polycode>
2. https://www.youtube.com/watch?v=xEU63D4zoLU&list=PLXgI6AEOr3lurN1feUMc0jErpJLCVGn5S&index=3&t=120s