2. Коли значення змінює початкове значення для іншого, з'являється ця кнопка, що дозволяє нам скинути його до значення за замовчуванням.
3. Коли ми зупиняємо симуляцію (кнопка зупинки на панелі попереднього перегляду), ми можемо побажати, щоб значення створеного нами моделювання відбувалися так само, як коли ми натискали відтворення (щоб повторювати таку ж ситуацію кілька разів), або, можливо, ми хочемо зберегти останнє значення (наприклад, щоб імітувати, що ми граємо декілька матчів поспіль). Якщо ми активуємо параметр "*Не зберігаємо*", зміни під час моделювання буде опущено, і коли симуляція буде зупинена, початкові значення будуть скинуті.

Лабораторна робота №2
Візуальна розробка елементів ігрової моделі

Анотація.

Лабораторна робота орієнтована на оволодіння студентами навичок візуальної розробки складних елементів ігрової моделі (руху та зіткнень ігрових об'єктів).

Лабораторна робота розрахована на 6 академічних годин і виконується малими робочими групами студентів.

Мета роботи:

Засвоїти навички і прийоми візуальної розробки елементів ігрового моделі засобами інструментарію WIMI5.

Обладнання:

Комп'ютерне обладнання ігрової лабораторії GameLab з встановленим інтернет-броузером Google Chrome.

Хід роботи.

1. Відкрити проект ігрового додатку, збережений за результатами виконання попередньої лабораторної роботи.
2. Визначити сценарії ігрової моделі, які потребують реалізації за проектом ігрового додатку.
3. Виконати декомпозицію сценаріїв на об'єкти логіки гри (входи, конектори, тригери, чорні скриньки тощо).
4. Розподілити об'єкти логіки гри між членами малої робочої групи.
5. Виконати розробку об'єктів логіки гри за допомогою інструментів WIMI5. Зберегти їх в ігровому проекті.
6. Протестувати працездатність сценаріїв та об'єктів ігрової моделі.
7. Обговорити між членами малої робочої групи та спроектувати метод реалізації руху та зіткнень ігрових об'єктів.
8. Спроектувати та реалізувати інструментами WIMI5 сценарії руху та зіткнень ігрових об'єктів.
9. Протестувати працездатність розроблених сценаріїв.
10. Інтегрувати розроблену ігрову модель в ігрове середовище, протестувати отриманий ігровий додаток.
11. Зберегти ігровий проект та опублікувати його в системі дистанційного навчання Moodle.
12. Продемонструвати розроблені елементи ігрового проекту викладачу.
13. Оформити звіт про виконання лабораторної роботи та подати його викладачу через систему дистанційного навчання Moodle.

МОДЕЛЮВАННЯ ЗІТКНЕНЬ

Наступні приклади можна знайти як шаблони на інформаційній панелі WiMi5. Просто натисніть кнопку "Клонування проекту" та виберіть будь-який з цих прикладів, щоб краще зрозуміти, як створити зіткнення.

У WiMi5 ви можете використовувати BlackBox *BoundingCollision* для виявлення зіткнень між об'єктами. Цей чорний ящик доступний в *2D* категорії вкладки blackboxes в *Logic Chart* панелі *редактора гри*.

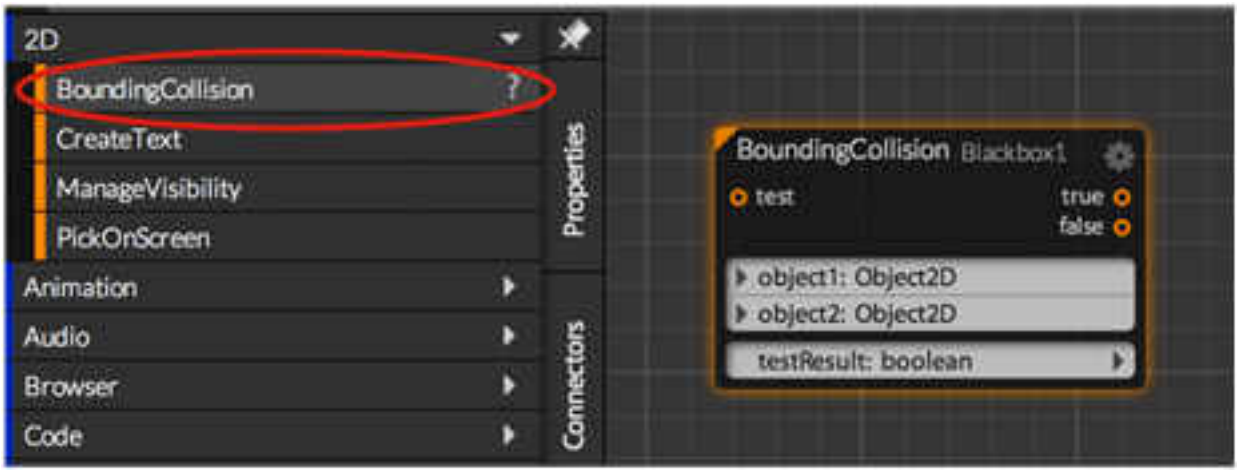

Рисунок 130. Чорний ящик зіткнень

Ви можете налаштувати цей чорний ящик. Якщо ви видалите цей чорний ящик на *логічну діаграму* та виберете її, ви зможете побачити вкладку властивостей для цієї чорної кухні. Ви можете вибрати тип зіткнення, який ви хочете виявляти: об'єкт з об'єктом або об'єктом з об'єктами.

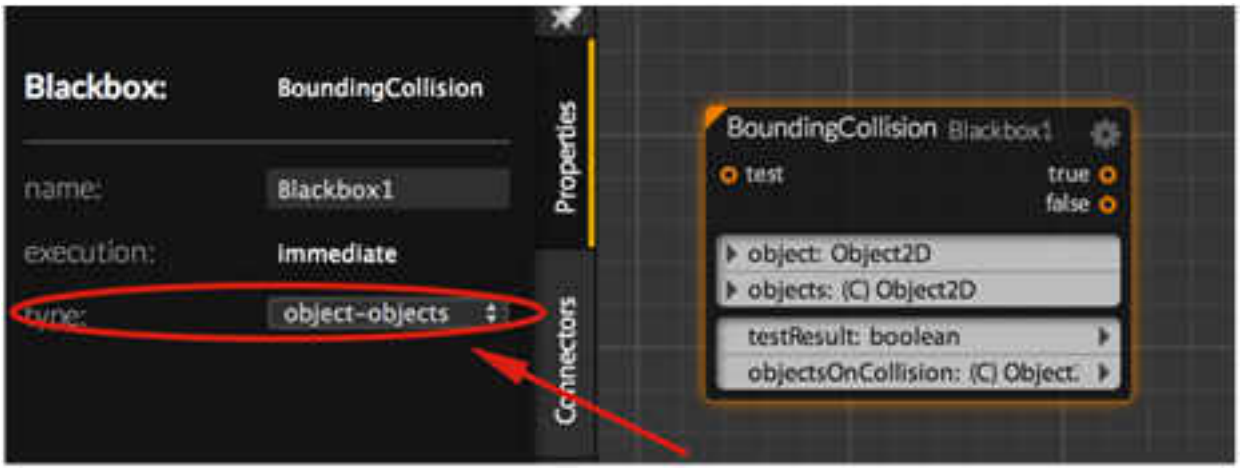

Рисунок 131. Вкладка властивостей чорної скриньки

Цей *чорний ящик* завжди має як вхідні параметри *Object2D* (Bitmap, Sprite та Text), на яких ми хочемо перевірити зіткнення. Перший параметр (*object1* або *об'єкт*) завжди є *Object2D,* а другий параметр

(*object2* або *об'єкти*) може бути іншим *об'єктом Object2D* або *об'єктом Object2D*, в залежності від налаштувань, які ви обрали для blackbox.

Коли ви активуєте цей чорний ящик, використовуючи *тестовий* запис, вихідний *істинний* буде запущений, якщо який-небудь з **блоків, що обмежує параметри,** стикається з іншим полем обмеження параметрів. І результат *ложь* буде запускатися, якщо не буде зіткнення. Ці обмежувальні коробки **не стикаються** з обмежувальними ящиками інших дітей *Object2D* в ієрархії. Ще однією цікавою властивістю цього чорно-бічного коду є те, що невидимі або **прозорі об'єкти також стикаються з іншими об'єктами**, тому ви можете створити недоступні для вилучення ділянки.

Крім того, *BlackBox BoundingCollision* виводить параметр *testResult*. Якщо тип зіткнення налаштовується як *objetc-objetcs,* виходом буде збірка, що містить об'єкти, які стикаються з параметром *object1*.

1. ВИЯВЛЕННЯ ЗІТКНЕННЯ МІЖ ДВОМА БІТНИМИ ЗОБРАЖЕННЯМИ

Метою цього прикладу є перетягування *растрового зображення* за допомогою миші та виявлення зіткнення між цим *растровим зображенням* та іншим. Коли буде виявлено зіткнення, ми будемо масштабувати зіткнену растровий малюнок для вказівки на зіткнення.

КРОК 1. СТВОРЕННЯ СЦЕНИ

Першим кроком є створення з *сценарієм редактора* сцени, що містить 2 *растрові зображення*. Завантажте свої растрові зображення в *Менеджер ресурсів,* а потім перетягніть їх на панель *SceneView* (рис. 132).

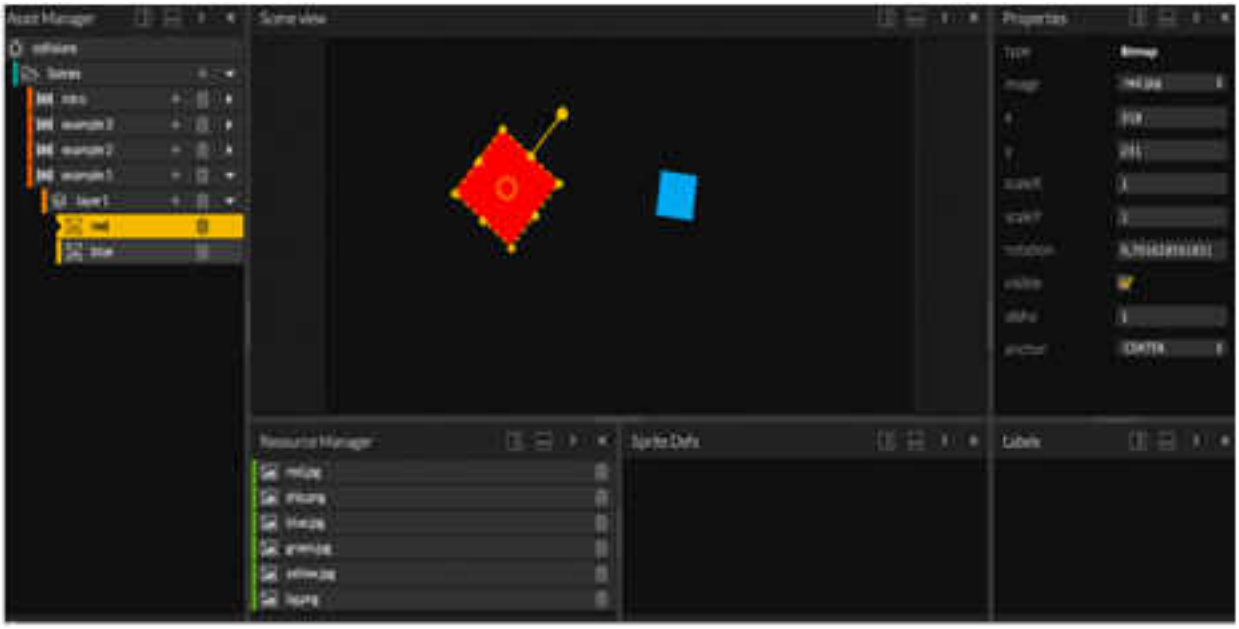

Рисунок 132. Створення сцени з двох об’єктів

КРОК 2. ПЕРЕТЯГУВАННЯ *РАСТРОВОГО ЗОБРАЖЕННЯ* ЗА ДОПОМОГОЮ МИШІ

Перетягніть *MouseListener* BlackBox (*LogicChart -> BlackBoxes* вкладка *-> Пристрої -> MouseListener)* в *діаграмі логіки*. Виберіть його і в *Властивості* вкладки зніміть *LeftButton* і виберіть *переїхав*.

На вкладці 2D на панелі *AssetManager* (рис. 133) ви можете вибрати один з імпортованих *растрових зображень* і залишити його в *LogicChart*. Ця дія автоматично створить Blackbox *BlackBox ActionOnParam*. Виберіть цей *чорний ящик* і на вкладці *Властивості* виберіть *setPosition* для властивості *Action*. Нарешті, перетягніть параметр *позиції BlackBox MouseListener* по параметру *позиції* Blackbox *BlackBerry ActionOnParam*.

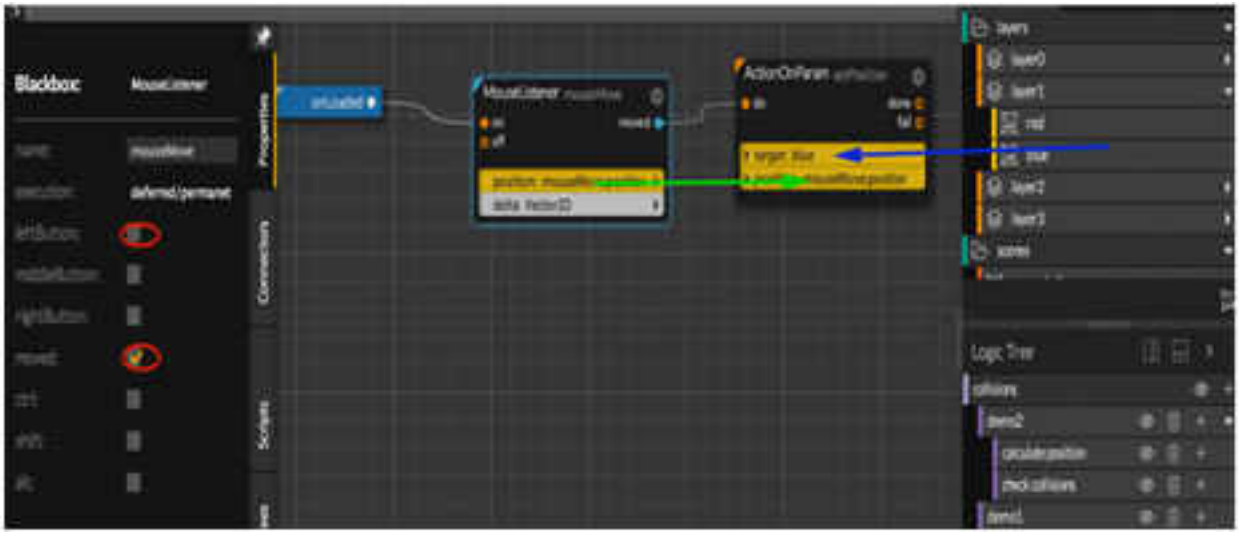

Рисунок 133. Вибір растрових зображень

КРОК 3. ВИЗНАЧЕННЯ ЗІТКНЕННЯ

Щоб виявити зіткнення між двома *Object2D,* вам просто потрібно перетягнути *BlackBox BoundingCollision* до *LogicChart*. Потім скиньте обидва *бітові зображення, які* ви хочете перевірити зіткнення з даними параметрів Blackbox. Якщо ви хочете перевірити зіткнення кожного разу, коли ви переміщуєте синій *растровий малюнок* (перетягнуте мишкою), ви повинні пов'язати *переміщений* вихід *MouseListener* з *тестовим* входом *BlackBox BoundingCollision*.

Порада: *якщо ви виберете створене посилання між чорними ящиками, ви зможете побачити його властивість замовлення дзвінка. Це властивість встановлює порядок викликів для посилань вихідного чорношету. Дуже рекомендую знати цей порядок дзвінків, щоб ви точно знали, як скрипти будуть виконані.*

Щоб візуально перевірити, чи виявлено зіткнення, ми будемо масштабувати червоний *бітовий малюнок,* коли він зіткнеться з синім. Ви можете перетягнути параметр *object2* (червоне поле)

з *BlackBox BoundingCollision* до будь-якого місця в *LogicChart*. Таким чином, *BlackBox ActionOnParam* буде автоматично створено за допомогою цього параметра, призначеного для *цілі*. Перетягніть його двічі, щоб ви могли встановити шкалу для обох можливих варіантів (зіткнення та відсутність зіткнення).

Виберіть створений *чорний ящик* і виберіть параметр *setScale* в його властивості *дії* (рис. 134). Ви легко можете призначити масштаб, натиснувши правою кнопкою вибору параметра *масштабу*. Після цього ви можете обрати створений параметр і встановити властивості *x* і *y* на 1.2. Вам доведеться повторити цей процес з іншою *панеллю BlackBox ActionOnParam* і встановити властивості *x* і *y* на 1.

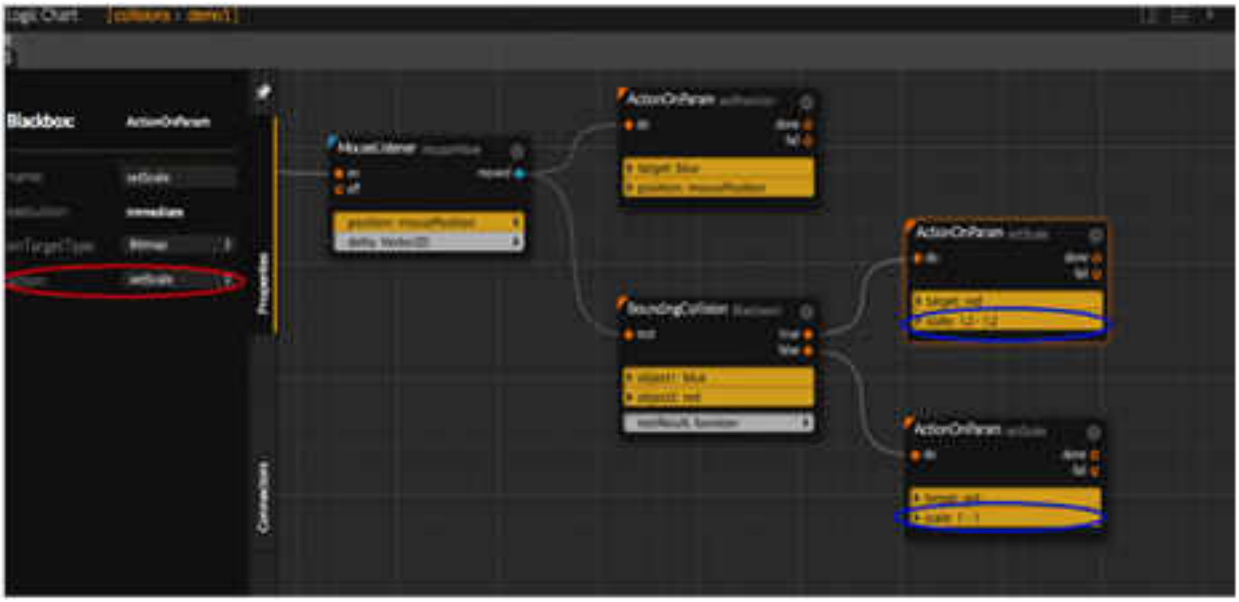

Рисунок 134. Визначення масштабу зображення

Тепер ви можете натиснути кнопку *відтворення* на панелі *попереднього перегляду* та перевірити свою роботу.

2. ВИЯВЛЕННЯ ЗІТКНЕННЯ МІЖ BITMAP І КОЛЕКЦІЄЮ BITMAP.

У цьому прикладі ми збираємося зробити квадратний стрибок з 4 стінками під час обертання.

\

КРОК 1. ПІДГОТОВКА СЦЕНИ.

Кожна з 4 стін буде *растровою*. Завантажте 4 растрові зображення для стін і поставте їх на сцену, як показано на малюнку нижче (рис. 135).

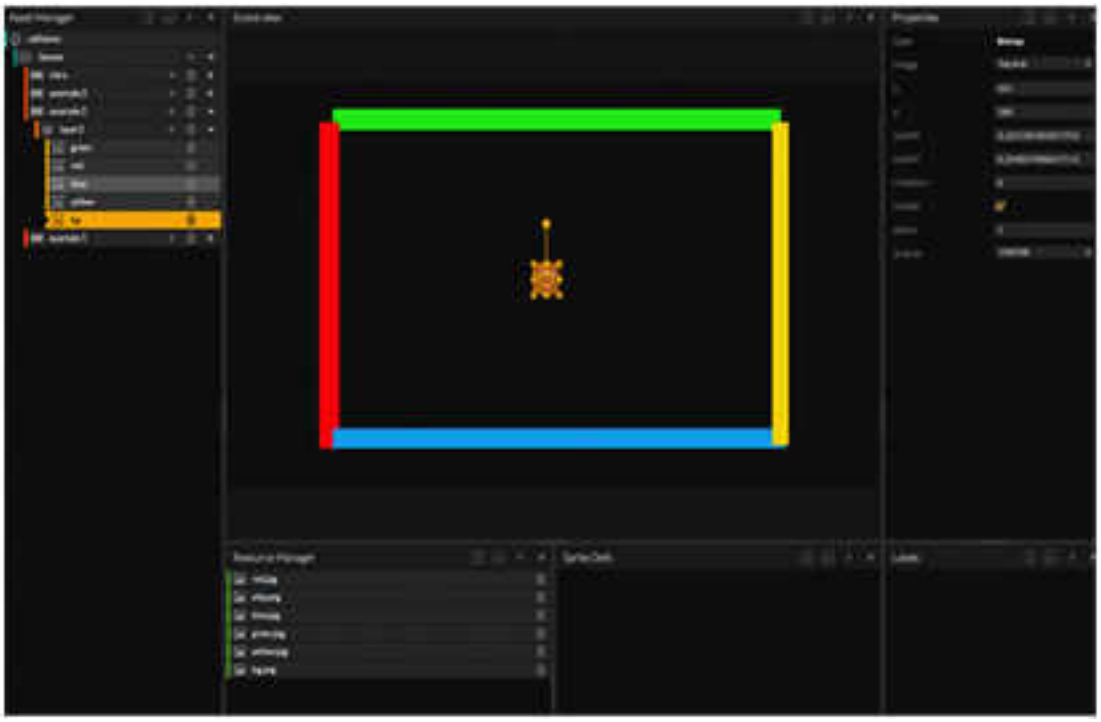

Рисунок 135. Завантажена сцена

КРОК 2. СТВОРЕННЯ КОЛЕКЦІЇ ЕЛЕМЕНТІВ, ЯКА ЗІТКНЕТЬСЯ

Після підготовки сцени потрібно створити *растрову* колекцію з 4 створеними стінами. Для цього потрібно перетягнути *AddItems* BlackBox (*LogicChart -> Blackboxes* вкладка *-> Колекції -> AddItems*). Вам потрібно перетягнути одну з *растровых* стін з вкладки 2D панелі *AssetManager* до параметрів *елемента 1* для *BlackBox AddItems* . Після цього новий параметр буде показано нижче першого. Тепер ви можете перетягнути наступну стіну на новий параметр і так далі. Потім вам потрібно клацнути правою кнопкою на параметрі *targer* і вибрати *Assign*, створюючи колекцію *растрового зображення,* яку ми будемо використовувати для додавання стін, налаштованих у параметрах.

КРОК 3. АНІМАЦІЯ СПЛИВАЮЧОЇ КОРОБКИ

Коли ми запустимо колекцію стін, нам потрібно оживити коробку, яка зіткнеться зі стінами. Для цього ви створюєте 2 різних швидкості на *X* і *Y* (так що ви можете самостійно змінювати напрямок на X або Y) і швидкість обертання. У кожному тактовому циклі ми застосуємо ці швидкості до положення та обертання коробки.

Перше, що нам потрібно зробити для анімації, - це певна дія, яка буде виконуватися в кожному тактовому циклі. Для цього ви можете використовувати *BlackBox KeepActive* (вкладка *blackboxes -> Stream -> KeepActive*). Цей чорний ящик має змінне число *на* тригерах. Після активації цих тригерів вони виконують відповідний вихід у кожному тактовому циклі,

доки *не* буде активовано вхідний сигнал, зупиняючи роботу чорної скриньки. *Чорний ящик* надає, у параметрі *deltaTime* , час, що минув від попереднього клаца. Більш того, для кожного доданого тригера виводиться *проміжок часу* параметр загальний проміжок часу після його активації. Вам слід призначити значення параметру deltaTime, натиснувши правою клавішею над чорною коробкою. Ми також будемо використовувати цей чорний ящик для активного виявлення зіткнень.

Для кращого розуміння та керування цим прикладом ви можете створити *сценарій,* який буде працювати з розрахунком позиції вікна в кожному тактовому циклі. Щоб створити *сценарій,* просто перетягніть кнопку «*Сценарій*» на вкладку «*Скрипти*» до «*Логічна карта*». Ви також можете створити *скрипт*, натиснувши на кнопку «+» в *скрипті* в *Logic Tree* панелі. Якщо ви двічі натиснете створений *сценарій,* ви отримаєте доступ до вмісту цього *сценарію*.

Якщо ви хочете створити анімацію з постійною швидкістю і з незалежністю від frameRate, вам слід знати, скільки часу минуло між кадром і наступним. Якщо ви помножите цей проміжок часу на постійне значення (швидкість), ви отримаєте необхідне значення для живих та постійних анімацій.

Найкращий спосіб зробити це - створити числовий параметр типу швидкості. Наприклад, перетягніть параметр з вкладки *Параметри -> Основні -> номер* в *LogicChart.* Це створить *BlackBox ActionOnParam* з *заданим числовым* параметром.

Ми створимо 3 *чорних ящиків,* як зазначалося раніше (*speedX, speedY* та *rotSpeed*). Потім клацніть на створені параметри, встановивши для значення 0,1 властивість *вмісту* для *speedX* і *speedY,* а 0,001 - для *rotSpeed*. Після того, як призначено швидкість, ви повинні перевірити, що обрана операція для *чорних ящиків* - це "*" (це операція за умовчанням для чисел) і перетягніть параметр *deltaTimed* у *BlackBox KeepActive* і *зменште* параметри *timesToMultiply* для кожного *actionsOnParam.*

Щоб перемістити растрове поле коробки, потрібно перетягнути растрове вікно з *об'єкта* з *AssetManager* і опустити його в *LogicChart.* У створеному *ActionOnParam* виберіть операцію *translateXY* . Потім потрібно провести перетягування результатів розрахунку швидкості *X* і *Y* *за* параметрами *перекладу X* та *Y.*

Повторіть той самий процес для обертання, використовуючи інший *ActionOnParam* з операцією *обертання*.

Для запуску *чорних ящиків* усередині цього *сценарію* потрібно створити вхідний *роз'єм,* який запускає роботу *сценарію (*вкладка «*Коннектори*» -> *«InputConnector»* перетягніть її до «*LogicChart»)*. Перейменуйте його, щоб *обчислити.*

Результат має виглядати наступним чином (рис. 136).

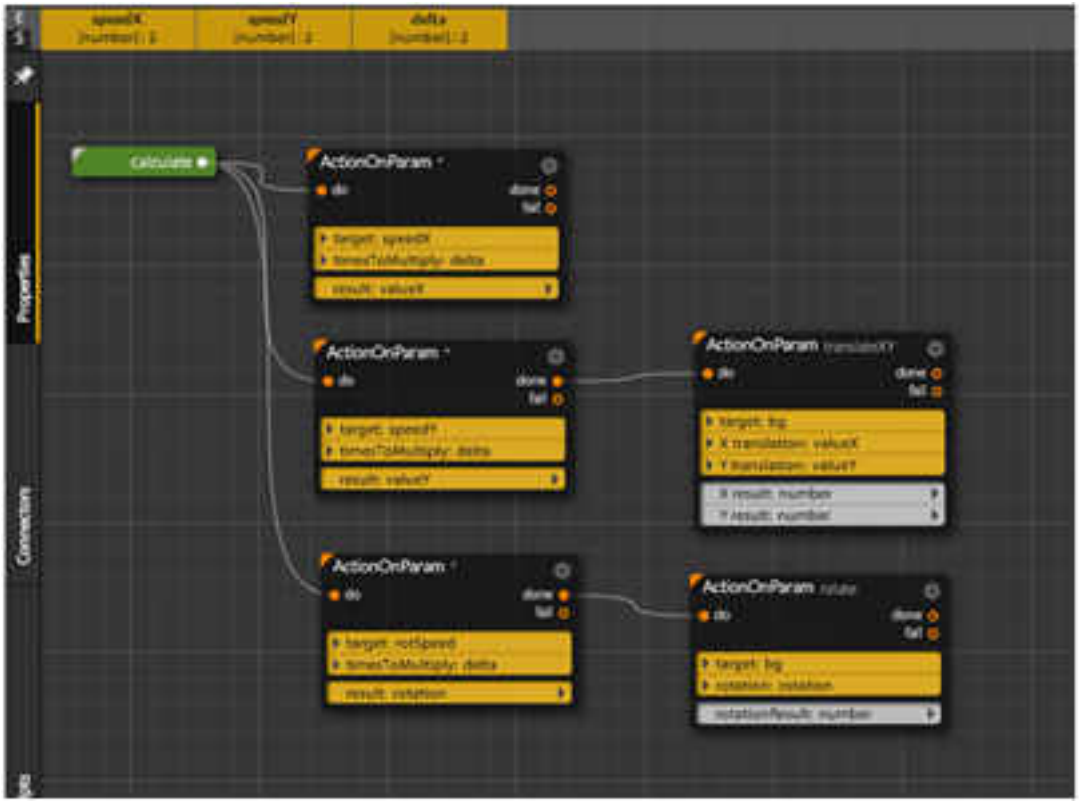

Рисунок 136. Результат побудови сценарію

Якщо ми з'єднаємо в кореневому *Script, KeepActive* BLACKBOX зі створеним *сценарієм* і натисніть на *Play* кнопці *попереднього перегляду* панелі ви повинні побачити, як коробка рухається (рис. 137).

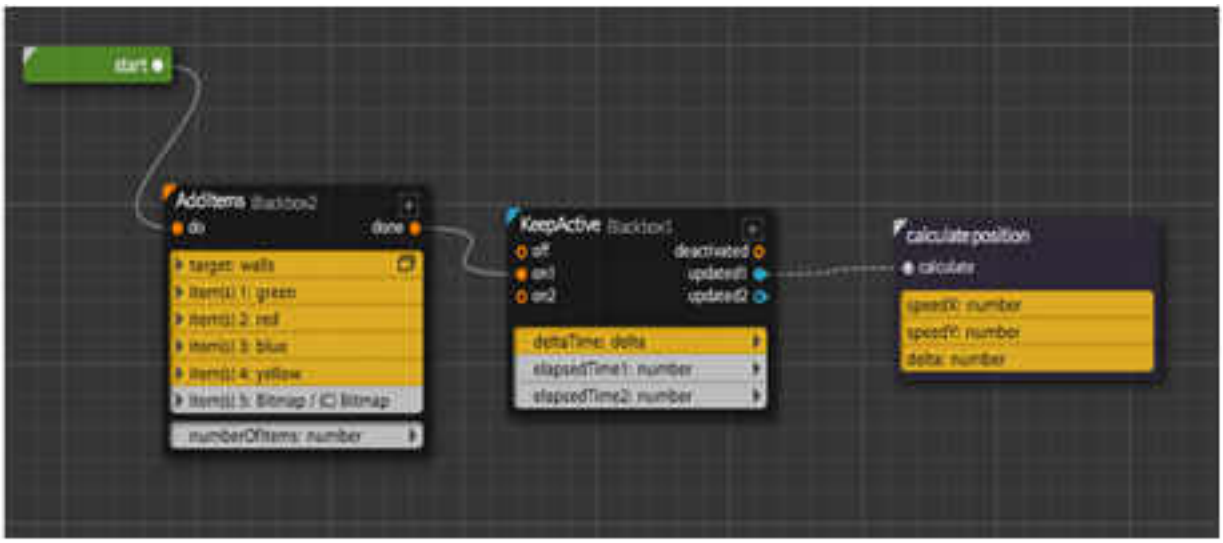

Рисунок 137. Гра Пошук серії слів

КРОК 4. ОБЧИСЛЕННЯ ЗІТКНЕННЯ ЗІ СТІНАМИ ТА ЗМІНА НАПРЯМКУ.

Тепер коробка рухається, але вона не стикається зі стінами. Ми збираємося розрахувати зіткнення з 4 стінами та змінити знак швидкості X або Y в залежності від стіни, з якою стикається коробка.

Як і раніше, ми збираємося створити новий *сценарій,* щоб мати кращий та більш організований візуальний сценарій. Ми створюємо *сценарій* і перейменовуємо його як *checkCollisions*. Усередині цього *сценарію* ми перетягуємо *чорну* коробку *BoundingCollision* та встановлюємо його властивість *muny object-objects*. Потім ми повинні скинути *растрове* вікно *коду* з *AssetManager* на параметр *об'єкта*. А в параметрі *objects* ми повинні скинути колекцію стін, створених на кроці 2 (рис. 138). Але цей параметр збірки знаходиться в кореневому *сценарії,* тому ми повинні повернутися до цього*Сценарій* та перетягніть цей параметр із *чорної скриньки AddItems* та *виділіть* його в новому *сценарії checkCollisions*. Тепер цей параметр доступний (у верхній панелі *LogicChart*), і ви можете випасти його на параметр *об'єктів*. Нарешті, ви повинні призначити об'єкт *Object2D*для вихідного параметра *ObjectsOnCollision* (рис. 139).

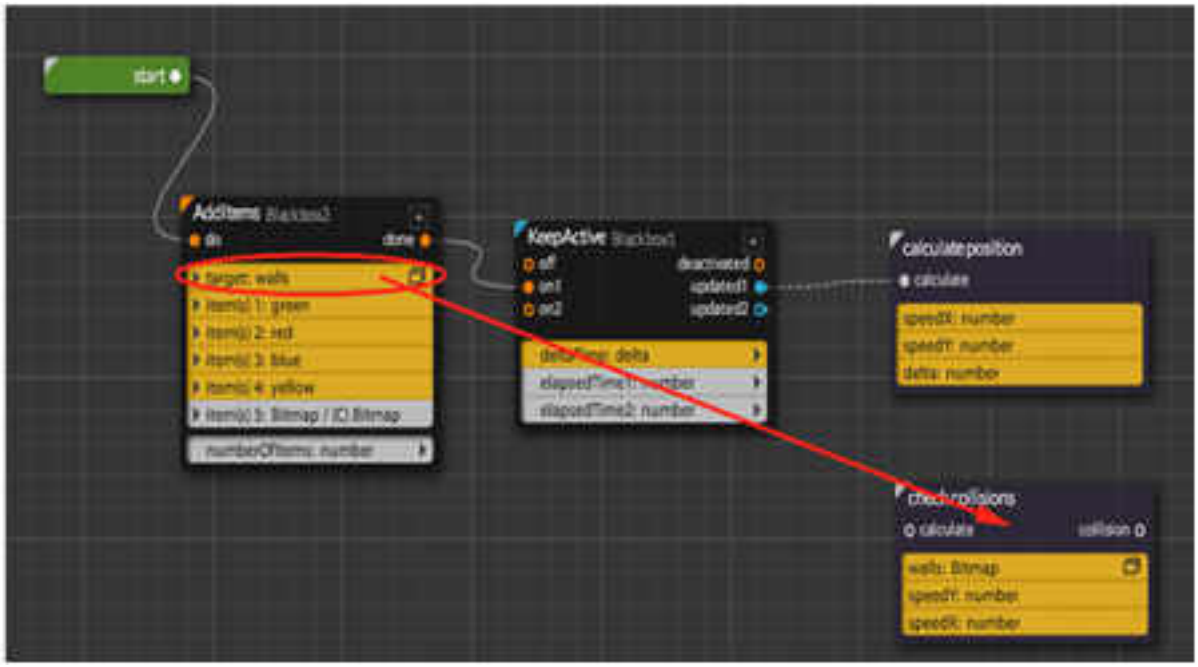

Рисунок 138. Гра Пошук серії слів

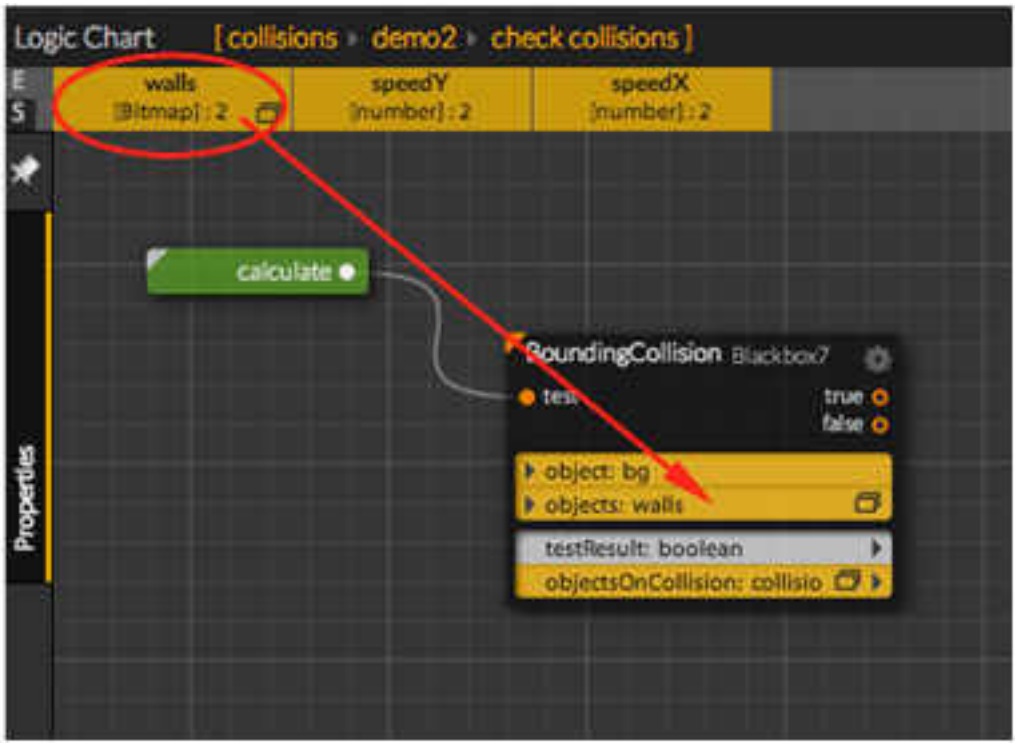

Рисунок 139. Гра Пошук серії слів

Тепер ми збираємося перевірити зіткнення з кожною стіною. Ми збираємося створити blackboxes *actionOnCollection*, перетягнувши 4 рази вихідну колекцію, створену раніше. Ми повинні встановити дії *містить* у створеній *ActionOnCollection*. Вам слід призначити кожну стінку растрові зображення для кожного другого параметра *actionOnCollection* (рис. 140). Як ви можете побачити спосіб виявлення зіткнення з стіною, запитаєте, чи ця стіна знаходиться в колекції вихідних *BlackBox BoundingCollision*.

Остання річ, яку ви повинні зробити, - це зміна знаку швидкості у X, якщо зіткнення - з лівими або правими стінами, або Y, якщо зіткнення з верхньою або нижньою. Щоб отримати доступ до цих параметрів швидкості, вам потрібно поділитися ними з *Script calculatePosition,* створеним на кроці 3. Вам слід отримати доступ до цього *сценарію* та скинути параметри швидкості для X і Y до верхньої панелі загальних параметрів у *LogicChart*. Потім ви повинні перетягнути їх, в кореневий сценарій, з *calculatePosition Script* до *checkCollisions Script* (рис. 141).

Рисунок 140. Сценарій перевірки зіткнення зі стіною

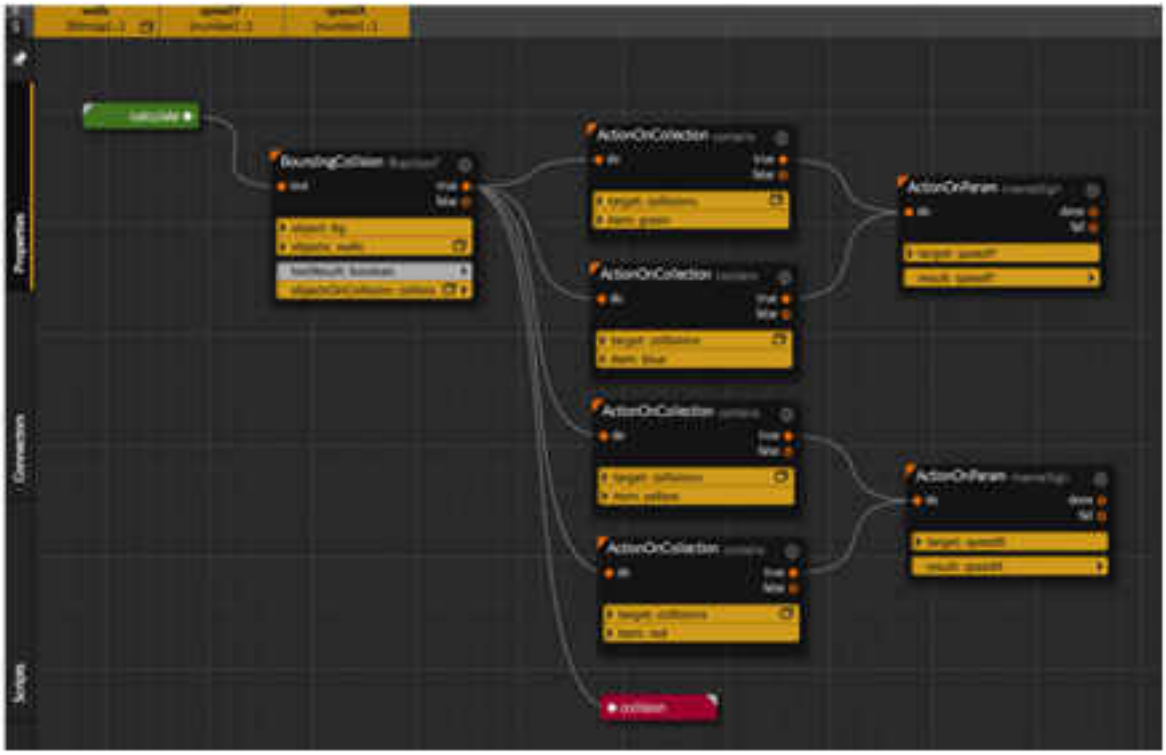

Рисунок 141. Сценарій зміни параметрів швидкості

Тепер ви можете пов'язати всі елементи (рис. 142).

Рисунок 142. Зв'язок елементів гри

Якщо ви запустите гру, ви побачите, що колізії правильно розраховані, але іноді коробка застряє в стінах. Це тому, що іноді, у час зіткнення коробка знаходиться всередині стіни. Щоб уникнути цього ефекту, ви можете зберегти позицію коробки (BlackBox *ParametersManagement -> Copy),* перш ніж знову її розрахувати, і, якщо зіткнення виявити, змініть знак швидкості, а також встановіть попередню позицію (рис. 143).

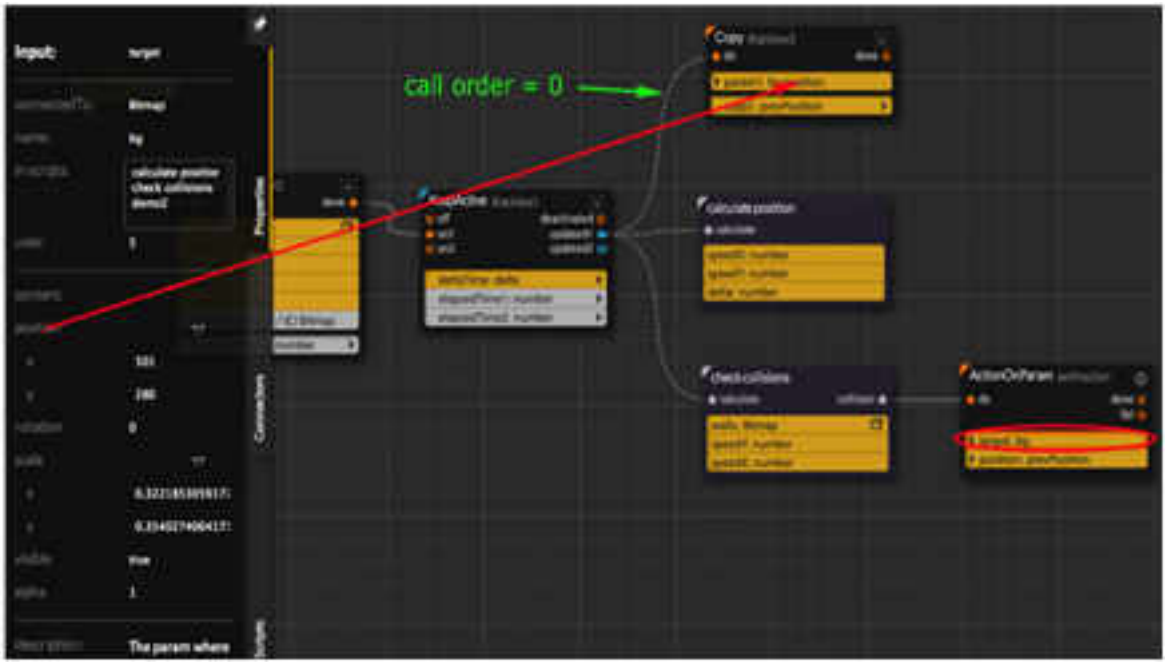

Рисунок 143. Сценарій дій при виявленні зіткнення

Ви можете отримати значення позиції, вибравши параметр, у якому відображається поле, і перетягнувши на вкладку властивостей позицію чорнила, яка їх потребує. Не забудьте встановити порядок *дзвінків* на "0" для зв'язку між *KeepActive* і *Copy* blackboxs.

3. ТОЧНІ ЗІТКНЕННЯ.

Одна з основних проблем із *BlackBox BoundingCollision* полягає в тому, що для розрахунку зіткнень використовується вся *вміщена* коробка *об'єкта2D*. Це означає, що якщо у нас є об'єкт з великою порожньою або прозорою зоною (велика кількість повітря), то зіткнення буде виявлено небажаним способом.

Для виявлення більш точних зіткнень ви можете створити кілька менших невидимих об'єктів і зависати їх у ієрархії об'єкта, який ви хочете обчислити зіткнення (рис. 144).

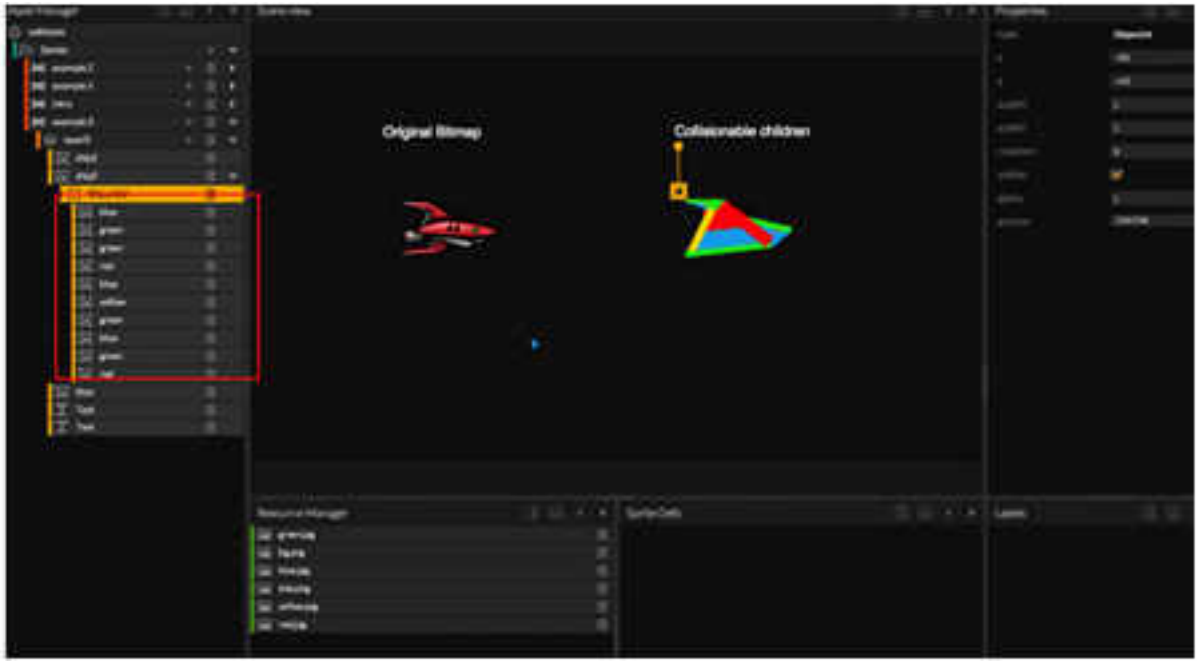

Рисунок 144. Створення невидимих об’єктів

Як ми вже згадували раніше, зіткнення також перевіряються з невидимими об'єктами, тому ми можемо встановити властивість об'єктів *видимим* значення *false*.

Тепер ми зробимо ті самі кроки, що і в прикладі 1 для першого космічного корабля (того, у котрого відсутні повішені об'єкти), і ми збираємося створити колекцію з зіткненнями об'єктів другого космічного корабля. Для цього ми можемо використовувати *getChildre* дію *ActionOnParam* Blackbox, використовуючи в якості *цільового* батьківського елемента з collidable *растрових зображень*. Ви можете побачити повний приклад на рис. 145.

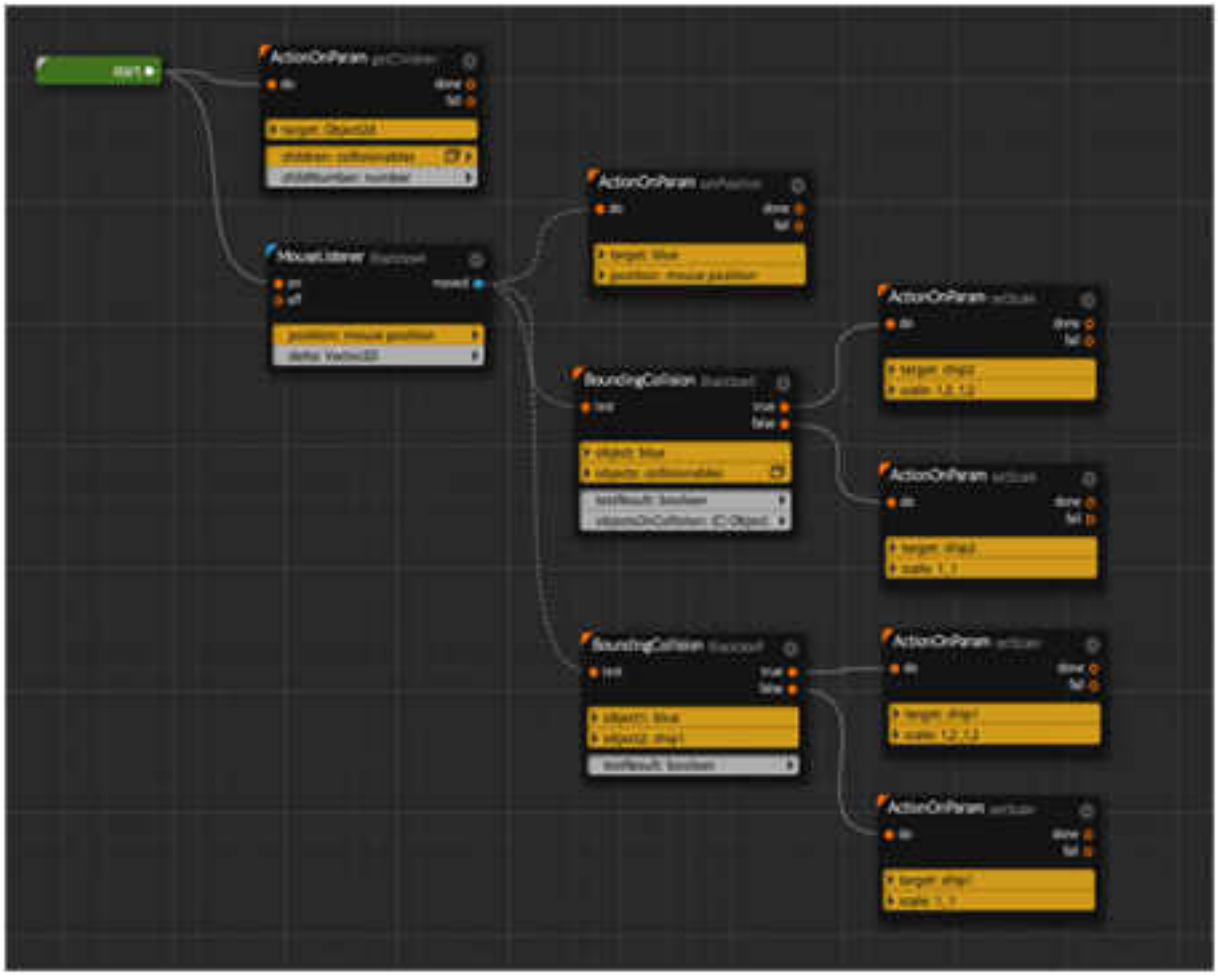

Рисунок 145. Повний приклад сценарію обробки зіткнення

Якщо ви натискаєте кнопку *відтворення,* ви можете перевірити відмінності між обома способами.

Нарешті, ви можете знати, що виявлення зіткнень є дорогим процесом. Можливо, ви захочете зробити ідеальні колісні коробки для всіх ваших елементів. Але це, ймовірно, буде більш ефективним, якщо ви спробуєте використовувати мінімальні зіткнення об'єктів для виявлення зіткнень.

Лабораторна робота №3
Монетизація та публікація ігрового проекту

Анотація.

Лабораторна робота орієнтована на оволодіння студентами навичок монетизації та публікації ігрових додатків.

Лабораторна робота розрахована на 6 академічних годин і виконується малими робочими групами студентів.

Мета роботи:

Засвоїти навички і прийоми монетизації, розгортання і публікації ігрових додатків засобами WIMI5.

Обладнання:

Комп'ютерне обладнання ігрової лабораторії GameLab з встановленим інтернет-броузером Google Chrome.

Хід роботи.

1. Відкрити проект ігрового додатку, збережений за результатами виконання попередньої лабораторної роботи.
2. Визначити елементи та об'єкти монетизації гри, які потребують реалізації за проектом ігрового додатку.
3. Розподілити об'єкти монетизації гри між членами малої робочої групи.
4. Виконати розробку об'єктів монетизації гри за допомогою інструментів WIMI5. Зберегти їх в ігровому проекті.
5. Реалізувати сценарії монетизації ігрового процесу та інтегрувати їх до ігрового проекту.
6. Протестувати працездатність ігрового додатку з інтегрованою монетизацією.
7. Виконанти інструментами WIMI5 розгортання ігрового додатку.
8. Опублікувати ігровий додаток на веб-сайті за вибором малої робочої групи та протестувати його працездатність.
9. Опублікувати ігровий додаток в системі Moodle спочатку в якості навчального ресурсу, потім в якості активності, перевірити його працездатність.
10. Зберегти завершений ігровий проект та опублікувати його в системі дистанційного навчання Moodle.
11. Продемонструвати ігровий проект викладачу.
12. Оформити звіт про виконання лабораторної роботи та подати його викладачу через систему дистанційного навчання Moodle.

СТВОРЕННЯ ВІРТУАЛЬНИХ ТОВАРІВ

Розглянемо, як створювати та використовувати Віртуальні товари, щоб монетизувати ігри HTML5 за допомогою покупок через додаток.

Для цього ми розглянемо можливості WiMi5 для здійснення покупок через додаток, і ми покажемо вам основний приклад створення різних віртуальних товарів.

Перше, що вам потрібно зробити - вирішити, що ви хочете продати. Ви могли б сказати, що все може бути продано. Ось кілька прикладів.

- Додаткові рівні.
- Об'єкти, що покращують ігровий досвід (більш потужна зброя або броня, імунітет, додаткова сила ...)
- Витратні предмети, що дають одноразові переваги в грі (бомби, додаткові життя, дорогоцінні камені / монети тощо).
- Пожертви для розробника.

Коли ви вирішите, які саме елементи ви хочете монетизувати в своїй грі, вам потрібно створити віртуальні товари у вашому проекті. Використовуйте інструмент інформаційної панелі для керування віртуальними товарами.

Щоб перейти до цього інструменту, потрібно перейти до налаштувань проекту, над яким ви працюєте (значок шестірні та налаштування). У налаштуваннях проекту виберіть вкладку " *Віртуальні товари"* (рис. 146).

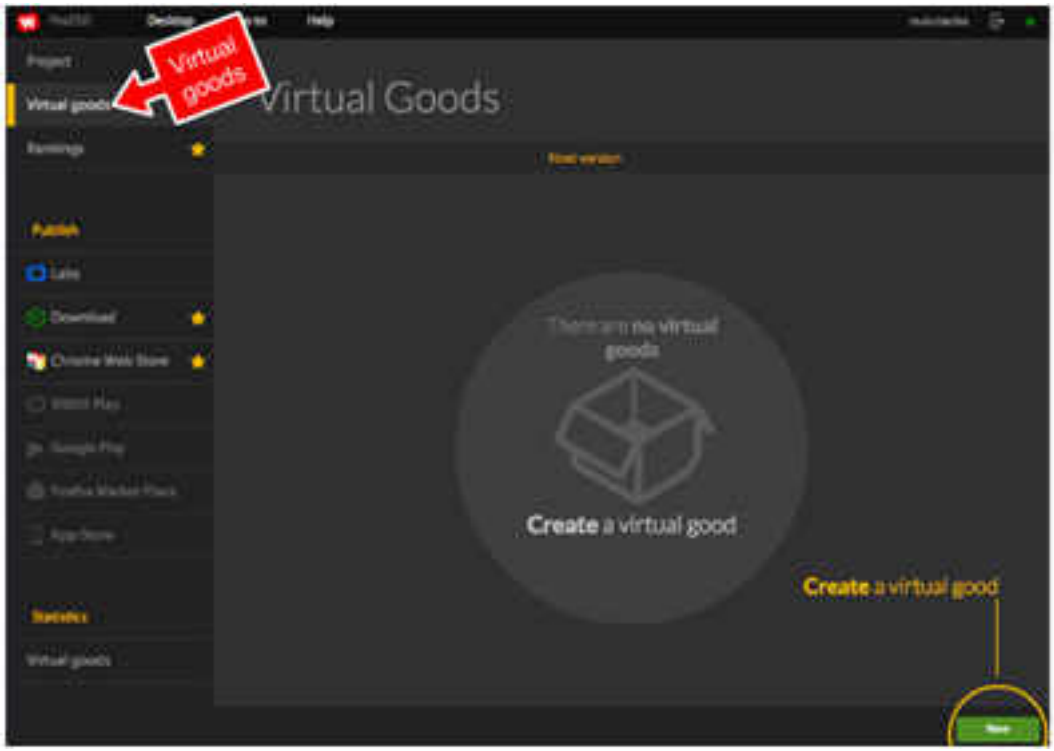

Рисунок 146. Вкладка Віртуальні товари

У нижньому правому куті ви знайдете кнопку, яка дозволяє створювати нові віртуальні товари до поточної версії проекту. Пізніше ми пояснимо, як управляти версіями та як вони впливають на віртуальні товари. Коли створюється новий Virtual Good, у списку віртуальних товарів з'явиться рядок (рис. 147).

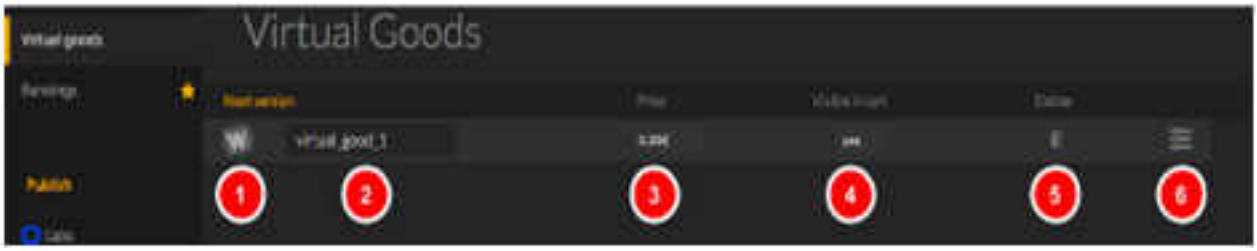

Рисунок 147. Рядок у списку віртуальних товарів

На зображенні ви можете побачити основні властивості, які можна змінити для кожного віртуального продукту, який ви створюєте:

1. Іконка для Віртуального Доброго. Це значок, який з'явиться у кошику для покупок. Натискання на нього дозволить нам вибрати зображення з нашого сховища зображень.
2. Назва віртуального блага. Це ім'я є внутрішнім і не відображатиметься в кошику для покупок. Він допомагає ідентифікувати віртуальний продукт у проекті та повинен бути унікальним у кожному проекті. Це впливає на всі версії Virtual Good.
3. Ціна, за якою ви хочете продати віртуальний товар. Зміна ціни вплине лише на поточну версію віртуального блага.
4. Видно в кошику. Цей прапорець дозволяє нам вибрати, чи є віртуальний продукт кошиком для покупок чи не відображається. Це корисно, коли ми хочемо запустити акцію або припинити продаж віртуального блага.
5. Видалити кнопку. Доступно лише в тому випадку, якщо немає інших версій віртуального продукту.
6. Додаткові опції. Натискання цієї кнопки відкриє ряд додаткових властивостей для віртуального блага (рис. 148).

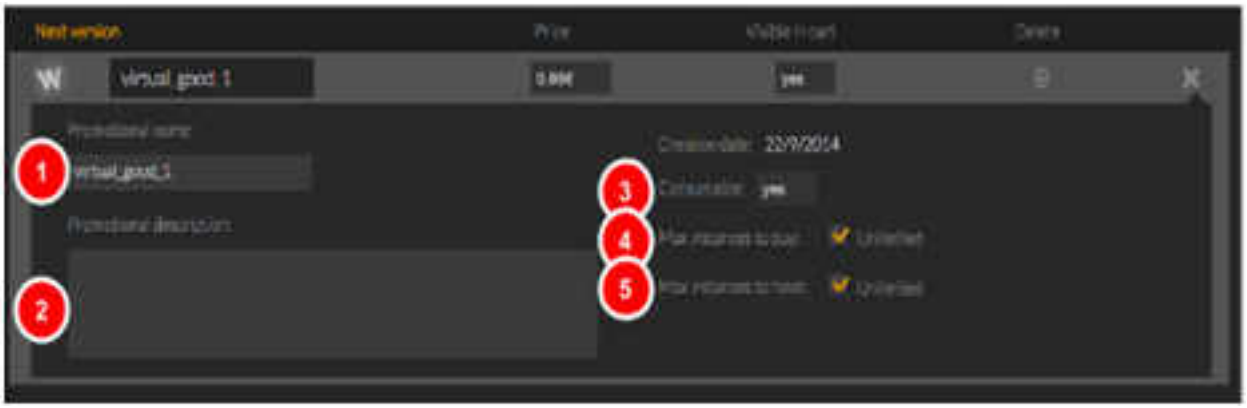

Рисунок 148. Налаштування властивостей віртуального блага

1. Рекламна назва. Це назва віртуального блага, як воно з'явиться у кошику для покупок.

2. Рекламний опис. Це опис віртуального блага, як воно з'явиться у кошику для покупок.
3. Це вказує на те, що продукт можна споживати, вираховуючи його з наявної кількості. Це типовий випадок для додаткових життів, монет, призупинених термінів, бомб і т. Д. Невитратний віртуальний благ буде той, який після придбання, він завжди доступний. Наприклад, додатковий рівень, унікальний об'єкт або деякі спеціальні можливості, придбані персонажем.
4. Макс екземпляри для покупки. Це максимальна кількість разів, коли віртуальний продукт можна придбати впродовж усього життя гри. Це може бути використано в основному для обмеження купівельної спроможності віртуального блага, що може зробити гру надто легкою. Наприклад, уявіть, що ви продаєте "ядерні бомби". Якщо ви дозволите занадто багато, щоб продати, це може зробити гру простою. Це значення не може бути більшим, ніж встановлене для наступної точки.
5. Максимальна кількість екземплярів. Це максимальна кількість разів, коли гравець може мати віртуальний результат у будь-який час.

ВЕРСІЇ ВІРТУАЛЬНИХ ТОВАРІВ

Щоб опублікувати гру в лабораторії, Chrome Web Store тощо, спочатку потрібно створити версію гри. Це робиться в редакторі гри за допомогою пункту меню *tools> deploy*. Натискання цієї опції генерує версію гри з певним ідентифікатором, і буде створено версію кожного віртуального товару, що міститься в грі в цей момент. Таким чином, віртуальний продукт може мати різні версії, де кожна версія - це налаштування цього віртуального продукту для цього конкретного розгортання.

Мета версії віртуальних товарів полягає в тому, щоб дозволити вам змінювати деякі властивості віртуального продукту, наприклад, ціну або його видимість у кошику для конкретної версії гри. Наприклад, ми могли б мати версію 0.1.10 гри, опубліковану в Веб-магазині Chrome, і працювати на версії 0.1.20 і змінити ціну віртуальних товарів у версії 0.1.10 ради просування в Chrome Web Магазин

Ви можете внести ці зміни в панель керування віртуальними товарами на інформаційній панелі. Якщо у вас є сформовані версії, внизу з'явиться спадне меню (рис. 149).

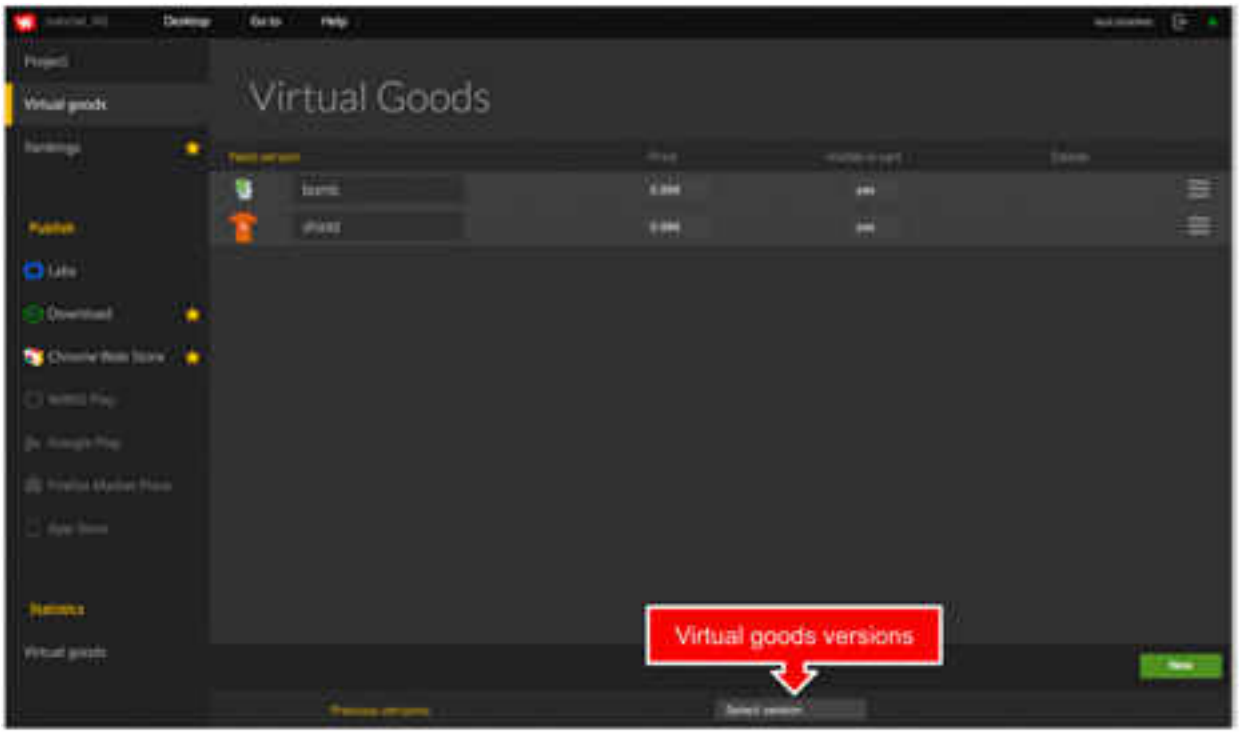

Рисунок 149. Спадне меню віртуальних товарів

Вибравши версію, ви побачите, як кожен з віртуальних товарів налаштовується в цій конкретній версії (рис. 150).

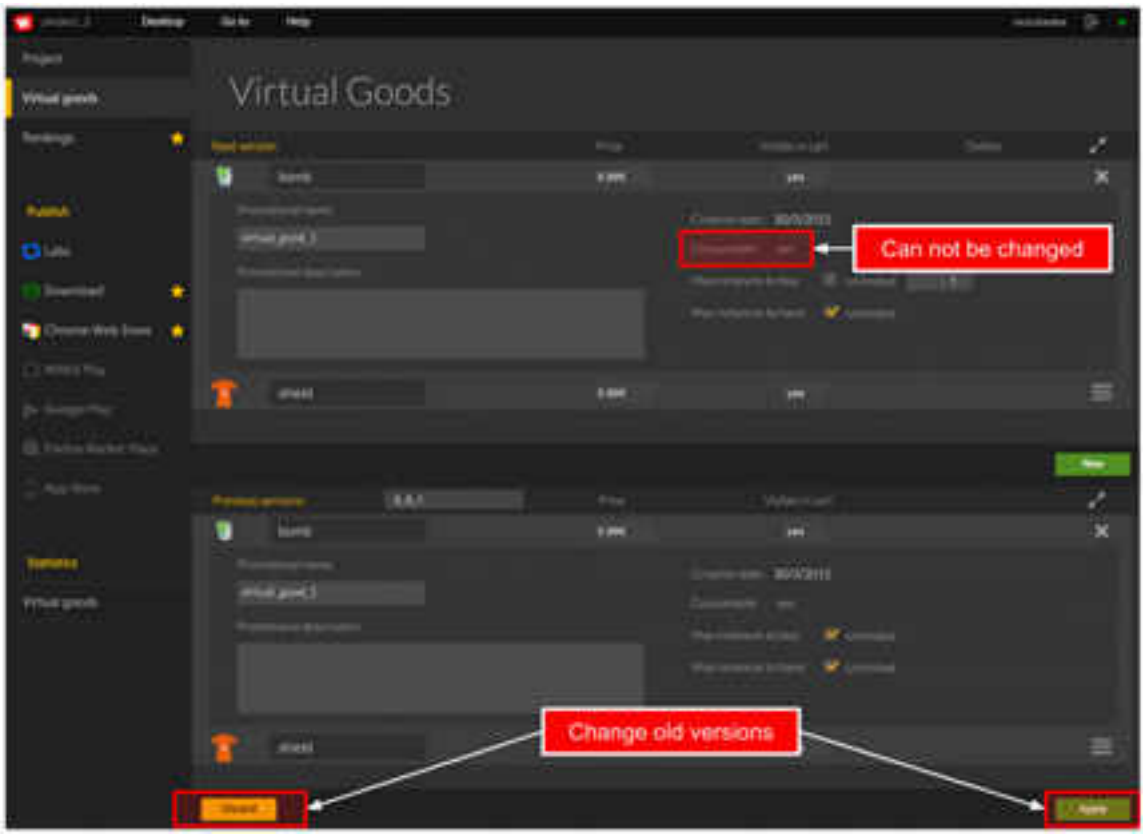

Рисунок 150. Налаштування віртуальних товарів

Всі властивості віртуального товару можуть бути змінені, за винятком *витратних* матеріалів поля, які повинні бути встановлені до створення першої версії VG. Кожного разу, коли будь-які властивості VG змінюються, з'являться кнопки *Apply* and *Discard*; вони просто внесуть зміни постійними або будуть ігнорувати їх, відповідно.

РОБОТА З ВІРТУАЛЬНИМИ ТОВАРАМИ

Коли віртуальні товари створені на інформаційній панелі, ви можете почати використовувати їх у редакторі. Для цього виберіть параметр «*Віртуальні товари» у* виборі будь-якої панелі (рис. 151).

Рисунок 151. Параметр Віртуальні товари

Натиснувши це, ви побачите віртуальні товари, створені з інформаційної панелі та деякі з їх властивостей (рис. 152).

Рисунок 152. Властивості віртуальних товарів

На додаток до зображення та імені віртуального продукту, кожна з піктограм:

1. Акумулятор: витратний або не витратний
2. Кошик для товарів і послуг: чи видно це там, чи ні.
3. Безліч символів: Максимальна кількість примірників віртуального блага може бути.

4. Поточні приклади: це значення служить для імітації кількості примірників, які ви повинні використовувати при попередньому перегляді гри. Ми можемо змінити його під час виконання попереднього перегляду, і при зупинці гру повернеться до значення, яке він мав перед тим, як *відтворити гру*. Це дозволяє імітувати різні стани віртуальних товарів, а також покупок і споживаних разів.

Для того, щоб використовувати віртуальні товари в рамках гри, у вас є серія чорних ящиків, згрупованих за категорією " *Віртуальні товари* " (рис. 153).

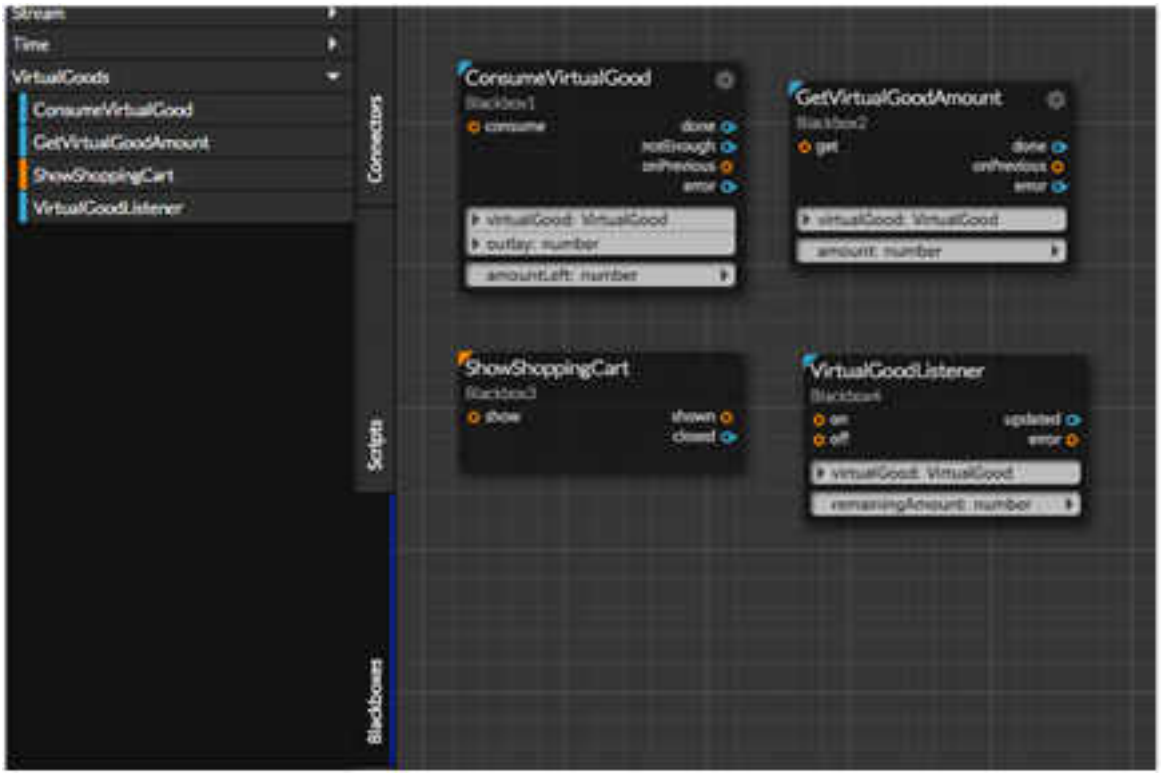

Рисунок 153. Чорні скриньки для віртуальних товарів

У вас є чотири різних чорних ящиків для керування віртуальними товарами в межах гри:

- **ShowShoppingCart**: це показує кошик для покупок і повідомляє через *закритий* вихід, закриття кошика для покупок. Він також повідомляє, що кошик був показаний. У редакторі, коли запускається попередній перегляд, буде показано симуляцію кошика для покупок і в остаточній грі (після натискання на команду deploy + test / publish) відобразиться остаточна кошик.
- **GetVirtualGoodAmount**. Це призводить до того, що користувач має певний віртуальний прибуток, залишаючи значення типу *number* у параметрі output *amount* . *Зроблено* роз'єм буде спрацьовувати , коли значення є. У випадку, якщо програвач не запустив

сеанс, або в петиції не було іншого типу помилок, виведення *помилки* буде активовано.

- **ConsumeVirtualGood**. Це споживає кількість екземплярів, зазначених у параметрі введення *витрати* віртуального *продукту*, зазначеного в параметрі *virtualGood*, і повертає кількість одиниць, які залишив гравець у параметрі *outputLeft*. Чотири різні вихідні роз'єми можуть спрацьовувати, запустивши цей чорний ящик залежно від результату операції. *Готово* спрацьовує, коли витратний матеріал споживається правильно. Якщо роз'єм *неефективний*, спрацьовує споживач, оскільки програвач не має достатньо великої кількості віртуального продукту. В обох випадках, коли ці виходи спрацьовують, *кількістьLeft* вихідний параметр доступний. Як і в першій "чорній" коробці, виведення *помилки* спрацьовує, якщо програвач не увійшов у систему, або в петиції не було іншого типу помилок.
- **VirtualGoodListener**. Цей чорний ящик має два стани, активні та неактивні. Для того, щоб активувати його, ви повинні запустити *на* вхідному роз'ємі. Після активації, він прослуховує всі зміни у кількості вказаного віртуального *продукту в* параметрі вхідного віртуального *ходу*. Коли зміна виявляється (споживається чи купується) у цьому віртуальному продукті, запускається *оновлений* вихідний роз'єм, який зберігає оновлену кількість екземплярів цього віртуального *продукту* в *reaningAmount*

Як ви можете бачити, в деяких *чорних ящиках* платформи (крім *GetVirtualAmount* і *ConsumeVirtualGood*) існує *вихідний* роз'єм. Цей коннектор з'являється в асинхронних чорних ящиках (докладніше про це пізніше) та в чорних ящиках, у яких доступний параметр *безпечного* налаштування (рис. 154).

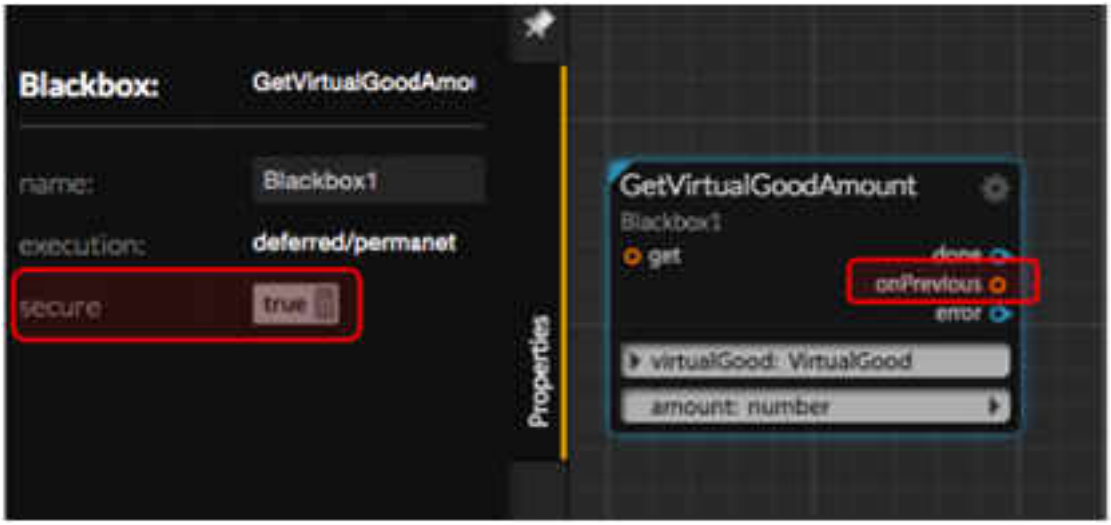

Рисунок 154. Вихідний коннектор асинхронної чорної скриньки

Безпечне властивість використовується, щоб уникнути дублювання декількох операцій на одному VG. Якщо ви встановите *захист* на *істину*, це означає, що ви не бажаєте починати нову операцію, доки попередня не буде завершена. Якщо це сталося, буде активовано *попередній* вихідний роз'єм.

Це може статися, якщо ми зробимо два послідовні клопотання дуже швидко. Якщо *безпечний* є *помилковою*, то всі клопотання будуть проводитися незалежно від того, є чи ні була відповідь на попереднє клопотання.

Примітка: асинхронна Blackbox - це той, вихідні роз'єми якого не запускаються миттєво, а лише в певний невизначений момент. Ці чорні ящики позначені синім трикутником у верхньому лівому куті чорної коробки, а асинхронні з'єднувачі також сині. Їхні зв'язки з іншими чорними ящиками є розривними.

ПРАКТИЧНИЙ ВИПАДОК

Для ілюстрації використання віртуальних товарів ми збираємося створити невеликий практичний приклад. У цьому прикладі ми хочемо монетизувати:

- Гравець може купити футболку. Цю сорочку можна купити лише один раз.
- Існує другий заблокований рівень, який можна розблокувати після придбання.
- Гравець може купувати фарбові бомби, щоб відзначити своїх ворогів, хоча він ніколи не може мати більше трьох бомб.

Таким чином ми створюємо 3 віртуальні товари наступним чином (рис. 155).

Як видно на зображенні, налаштування для віртуального продукту, який називається *рівнем,* і ті, що є для *футболки* , абсолютно однакові. Ці два віртуальні товари не є витратними, і вони можуть мати лише один раз. З іншого боку, *фарби Бомбові* товари є витратними, але ми обмежили їх максимальну кількість екземплярів до трьох. Номер, який можна придбати, необмежений, а це означає, що ми залишаємо відкритим двері, щоб купувати більше, коли ми споживали ті, що ми маємо.

Гра починається зі сцени для вибору рівня, натискаючи відповідні кнопки, а іншу, що відкриває корзину для покупок (рис. 156).

Коли ви вступаєте в гру, сцена активована за умовчанням, і ми створимо наступний скрипт (рис. 157).

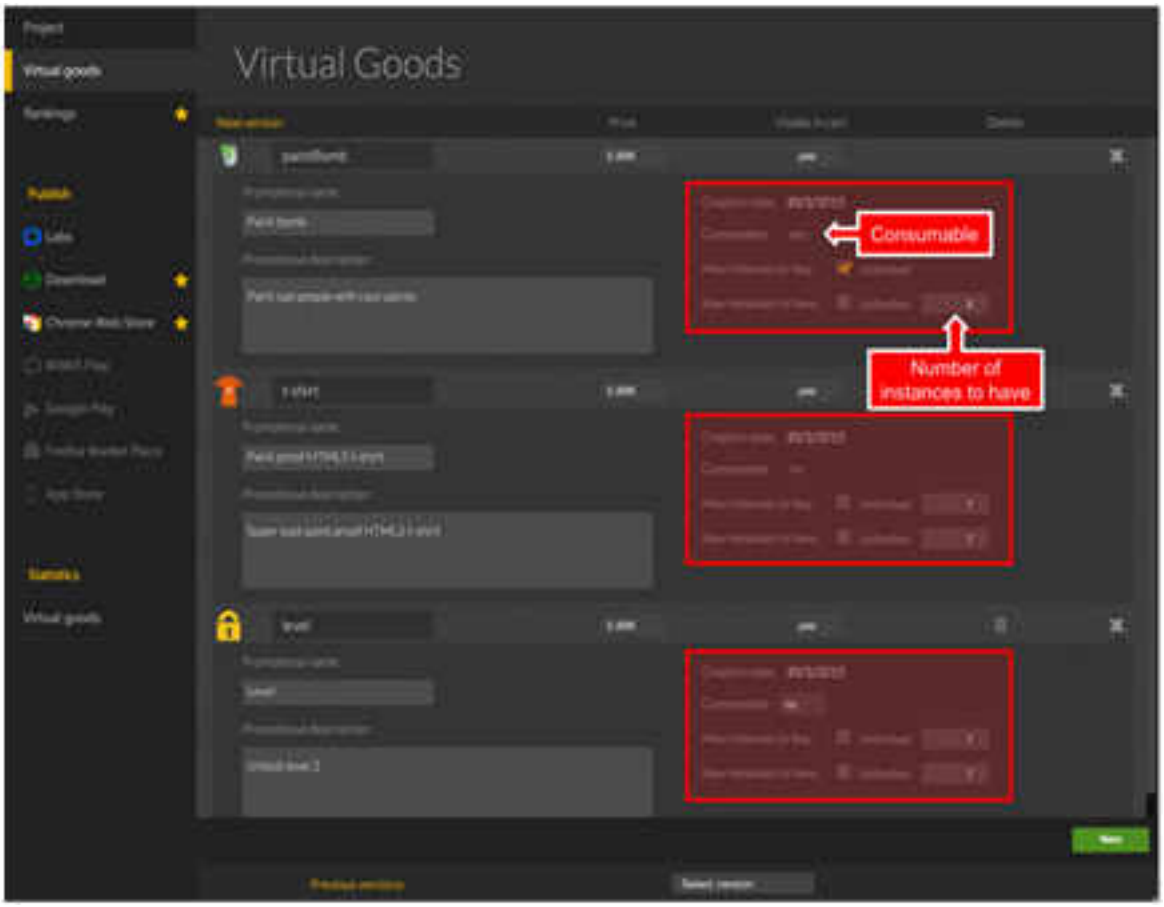

Рисунок 155. Утворення віртуальних товарів

Рисунок 156. Корзина для покупок

Рисунок 157. Скрипт активації сцени

1. Перше, що ми повинні робити при запуску гри, - привласнення поведінки кнопок (спрацьовування) за допомогою *SpriteAsButton*. Нам потрібно встановити анімацію відключення (*OnDisableAnim*) для кнопки другого рівня, до якої ми призначили рамку анімації з замок, що вказує на те, що цей рівень заблоковано. Таким чином, ми ініціалізуємо кнопки для рівня 1 з *активним* входом, а рівень 2 - з *вимкненим* входом. Ми створили *Рівень* під назвою "play", який відповідає за керування самою грою, починаючи відразу після того, як ми пройшли через сцену. Коли ми натискаємо одну з кнопок, ми скопіювати посилання на сцену, яку ми хочемо завантажити, до *сцени*-тип вхідного параметра ми створили в цьому «грати» *рівні* .
2. Далі ми поцікавимося про віртуальний *продукт другого рівня* , і якщо ми придбали один примірник у цьому випадку, ми активуємо кнопку, яка дає доступ до другого рівня. Ми також активуємо *BlackBox VirtualGoodListener,* який повідомлятиме нам про будь-які зміни, внесені до віртуального продукту *2 рівня*. Необхідно мати на увазі, що для порівняння, яке буде проведено правильно, змінна *суми, яка* використовується як параметр у трьох чорних ящиках, повинна бути однаковою.
3. Коли ви натискаєте кнопку "Кошик для покупок", ви відкриєте кошик для покупок за допомогою *ShowShoppingCart*

Гра в основному прослуховує натискання клавіші B, і коли вона є, якщо наявні фарбувальні бомби, то він виконує анімацію вибуху на одному

бюрократі (2), що є на екрані (рис. 158). Він також повідомляє кількість бомб, доступних через текст (1), і дає можливість вийти з кошика для покупок, щоб здійснити іншу покупку (4). Якщо гравець вже купив футболку, символ (3) буде показаний футболкою, яка відрізняється від стандартної.

Рисунок 158. Прослуховування клавіш у грі

Сценарій, який обробляє цю логіку гри, полягає в наступному (рис. 159).

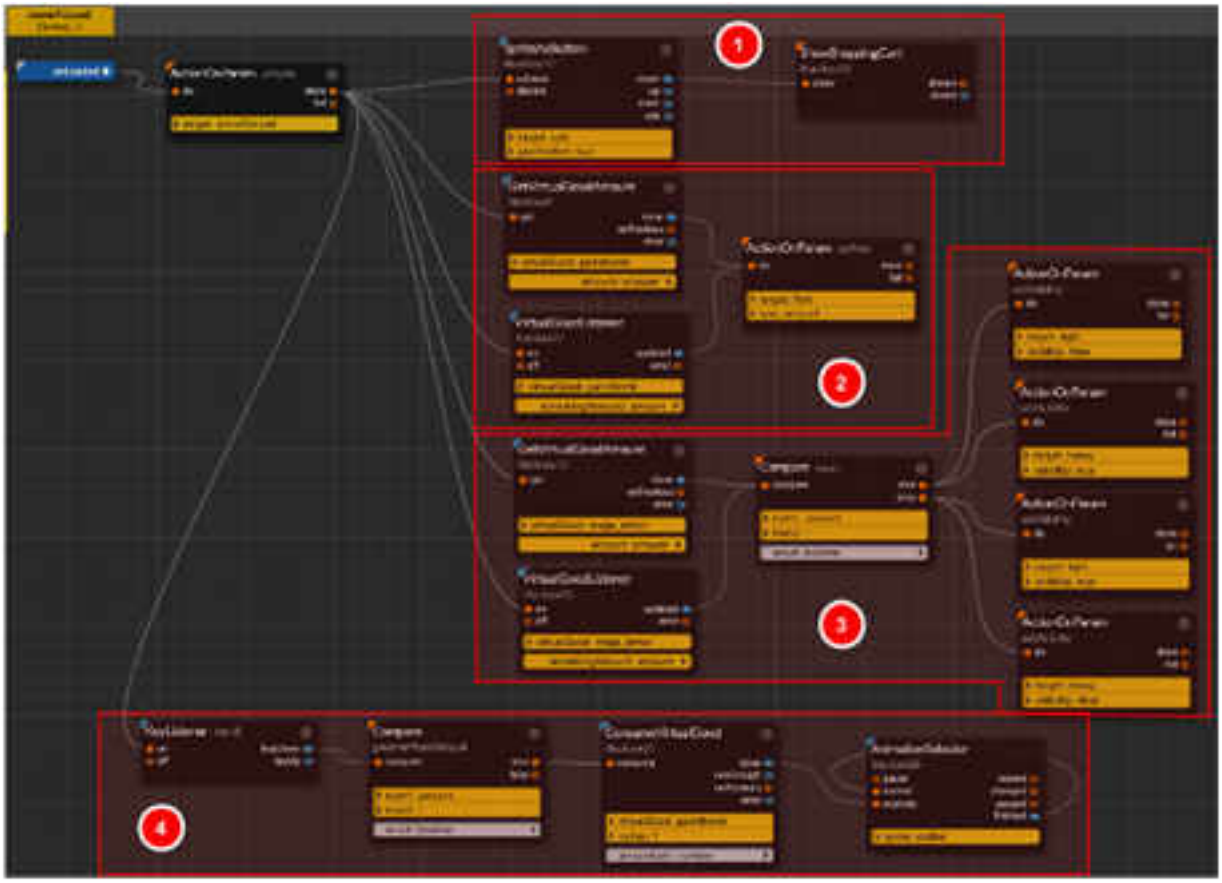

Рисунок 159. Гра Пошук серії слів

1. У цій частині ми створимо кнопку, щоб показати кошик для покупок, точно так само, як і на попередньому рівні.
2. Потрібно спочатку запитати (*GetVirtualAmount*) про кількість доступних бомб, щоб показати їх гравцеві через екранний текст. Ми також будемо слухати зміни у кількості бомб, які має програвач (*VirtualGoodListener*).
3. Ми також повинні робити те ж саме для футболки гравця, запитуючи, чи має вона її, і підписатись на зміни суми. Ми могли б зробити це і останнє в унікальному випадку, коли відповідь на *BlackBox GetVirtualGoodAmount* була 0, але для наочності скрипту це було зроблено таким чином. Якщо у нас є футболка доступна, ми зробимо футболку відома на символі, і якщо ми її не матимемо, ми зробимо футболку за замовчуванням видимою (зауважте, що трохи про те, що робити за замовчуванням t - що ми могли б зберегти, встановивши в редакторі сорочку за замовчуванням, але завжди є ідеєю ініціалізувати елементи усередині сцени).
4. Ми додали *KeyListener*, щоб прослухати преси клавіші B. Якщо це буде натиснуто, ми перевіримо, чи існують бомби, і якщо ми отримаємо позитивне твердження, ми скажемо про те, що бомба споживається, і ми викличемо анімацію вибуху (рис. 160).

Рисунок 160. Анімація вибуху

Другий рівень буде нічим іншим, ніж та сама сцена з різним фоном. Ми можемо створити загальний *шар* між обома сценами, щоб розмістити всі загальні елементи (символ, маркер, кнопки тощо) і, таким чином, робити логіку сценарію набагато простішою.

Статистика віртуальних товарів

На інформаційній панелі є інструмент для контролю над доходами, отриманими від ігор, які використовують віртуальні товари. Щоб використовувати цей інструмент, нам потрібно мати гру, яка використовує віртуальні товари та яка була опублікована (яка була *розгорнута* в

редакторі). Щоб переглянути статистику віртуальних товарів, вам потрібно перейти до налаштувань проекту на інформаційній панелі та вибрати вкладку *віртуальних товарів* у розділі *статистики* (рис. 161).

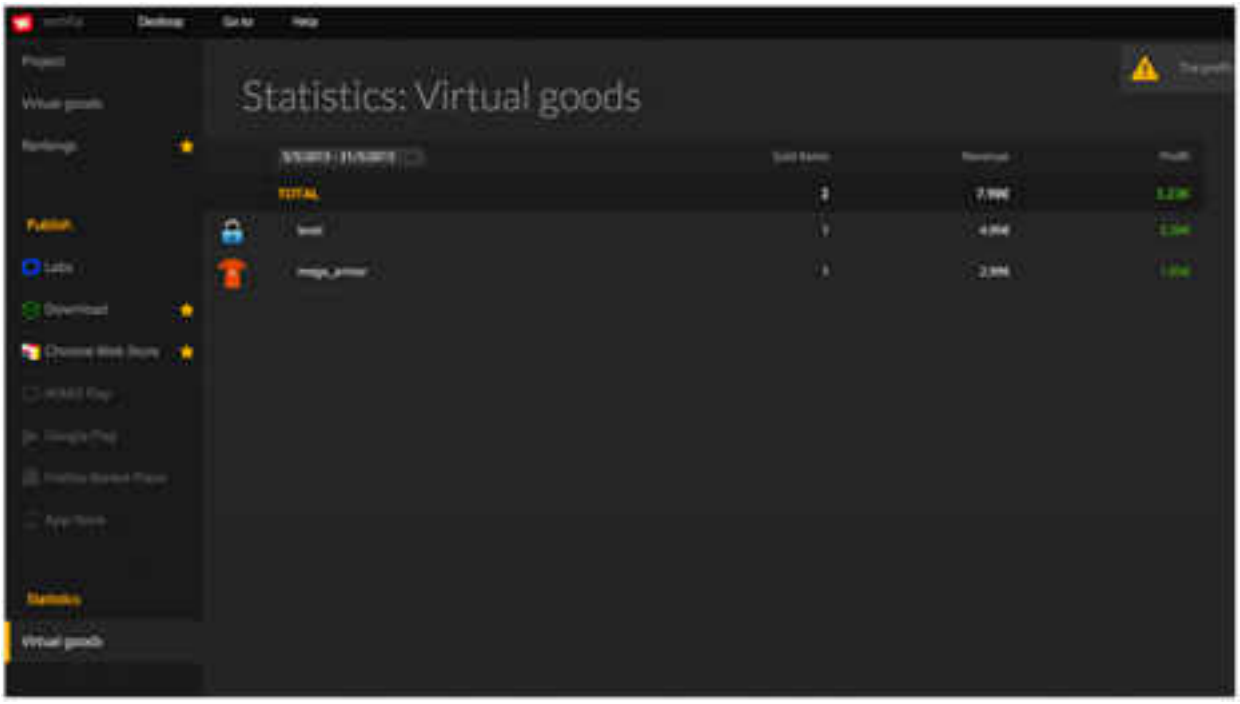

Рисунок 161. Розділ Статистики віртуальних товарів

Ви знайдете на цій вкладці список всіх віртуальних товарів, які містить проект, та їхні доходи:

- **Продані предмети**: це кількість віртуальних товарів, які були продані.
- **Дохід**: це загальний дохід від цього віртуального блага.
- **Прибуток**: кожен з платформ публікації має власні правила та комісії щодо продажів, створених додатками. Це значення показало кінцевий прибуток користувача від продажу цього віртуального блага. Ви повинні мати на увазі, що це число є приблизним.

Також є селектор дат, який дозволяє переглядати дохід від віртуальних товарів у різні періоди часу (рис. 162).

Інтеграція гри на веб-сайт

Прикладом гри є те, що створено за допомогою редактора онлайн-ігор WiMi5. Ми побачимо, як інтегрувати цю гру в кілька менеджерів контенту (CMS) та інструментів для створення блогів. Зокрема, ми зосередимось на:

- WordPress
- Joomla
- Blogger

В основному ми повинні вставити "iframe" всередині вмісту, який ми хочемо, у кожному з варіантів, на які будемо дивитися.

Рисунок 162. Селектор дат

"iframe" буде містити URL-адресу гри, яку ми хочемо інтегрувати на сайт, та інші параметри, щоб персоналізувати його зовнішній вигляд відповідно до наших уподобань.

Майже у всіх варіантах, які ми розглянемо, інструменти керування вмістом мають редактор, який має два параметри перегляду того, що створено.

З одного боку, є режим WYSIWYG, який дозволяє переглянути остаточний вигляд вмісту. З іншого боку, є режим редагування HTML-коду, де ви можете безпосередньо вставляти теги HTML, як у випадку з "iframe".

Для прикладу ми використаємо одну з освітніх ігор, доступних на educa.land.

На сторінці гри 'Match additions' (рис. 163) ви можете переглянути та скопіювати код 'iframe', необхідний для інтеграції гри в різні менеджери вмісту та інструменти створення блогів, які ми будемо переглядати.

Код є наступним:

<iframe id = "playframe" src = "https://educa.land/games/matchaddition/0_0_11/" frameborder = "0" scrolling = "no" width = "100%" height = "576"> </ iframe >

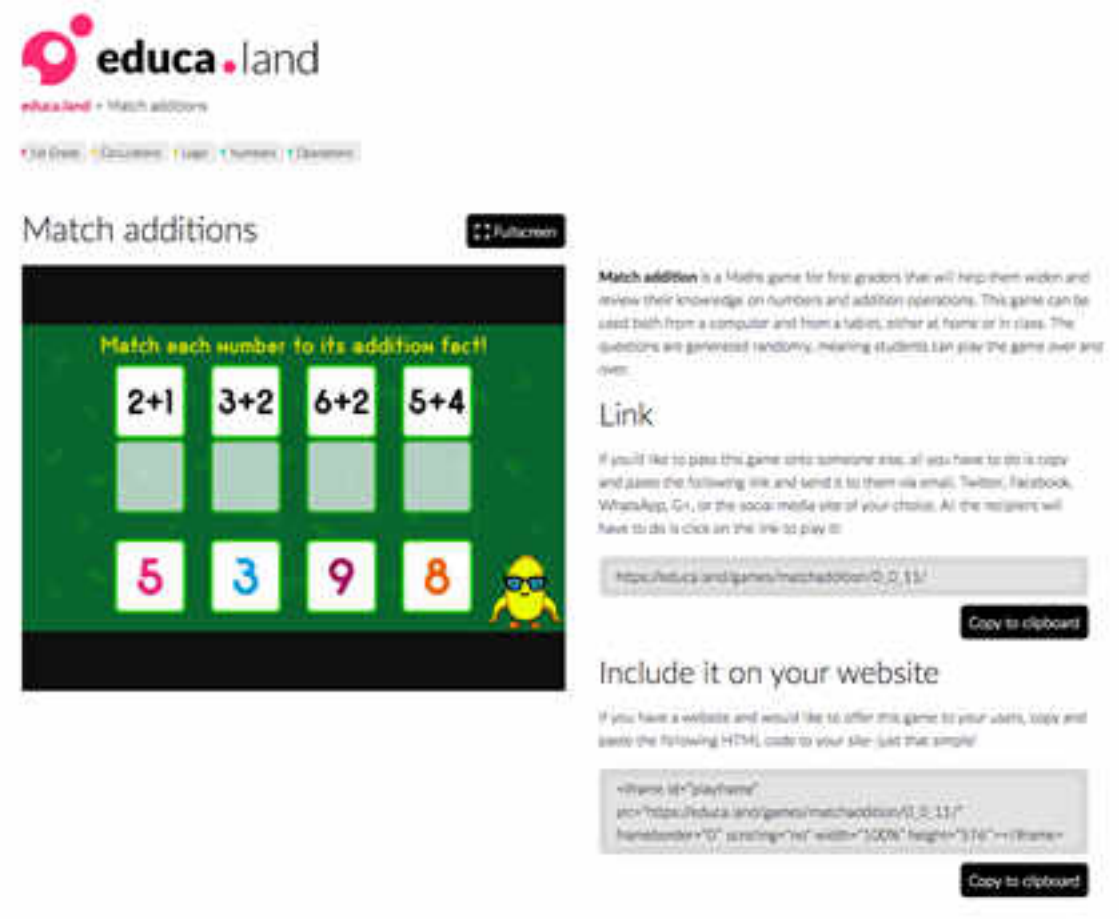

Рисунок 163. Сторінка гри Match additions

ІНТЕГРАЦІЯ ГРИ У WORDPRESS

Щоб інтегрувати гру HTML5 з "iframe" в WordPress, ми повинні забути про використання служби WordPress.com, оскільки вона не дозволяє використовувати "iframes".

Натомість ці інструкції написані для веб-сайтів, створених за допомогою WordPress.org, тобто сайтів, створених користувачем, що встановили цю систему керування вмістом.

У цьому випадку інтеграція гри HTML5 дуже проста. Все, що вам потрібно зробити, це скопіювати код "iframe" гри, який ми отримали на сторінці 'Match additions' на educa.land (рис. 164).

Рисунок 164. Вставка коду "iframe"

Потрібно вставити його в вікно редагування статті або сторінки, яку ми створюємо за допомогою WordPress. Ви повинні бути впевнені, що використовуєте режим "Текст" у текстовому редакторі, щоб правильно вставити код.

Потім просто опублікуйте (рис. 165), і все.

Рисунок 165. Опублікування гри

ІНТЕГРАЦІЯ ГРИ В JOOMLA

Щоб інтегрувати гру в Joomla, ми збираємося знову використовувати приклад гри з educa.land, Match Updates. Ми збираємось скопіювати фрагмент коду, який міститься в розділі "Включити на свій веб-сайт".

Відкрийте адміністратора веб-сайту, створений Joomla, і додайте нову статтю (рис. 166).

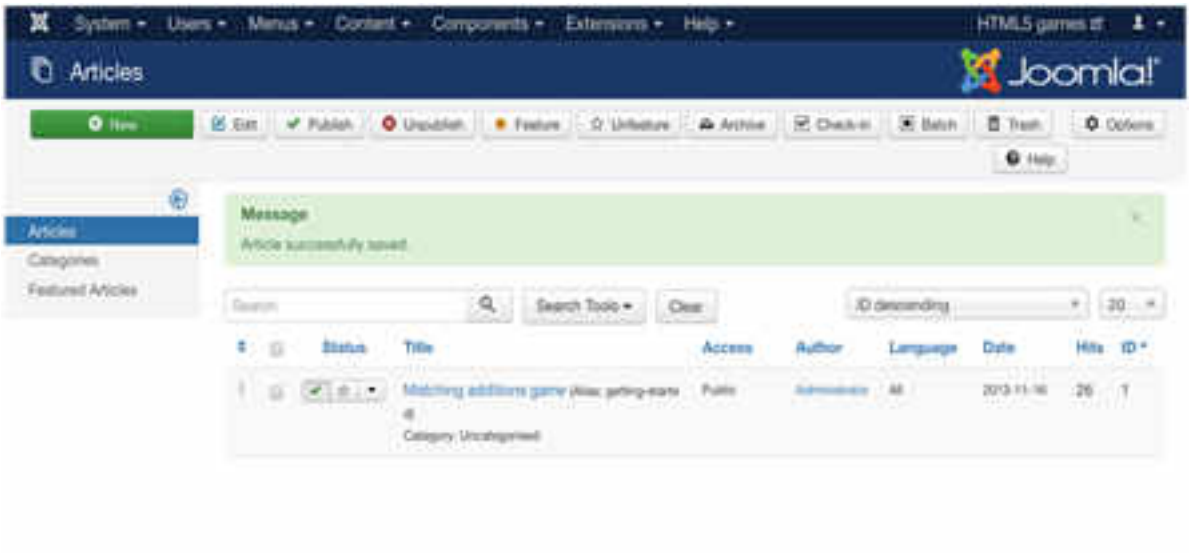

Рисунок 166. Додавання нової статті

Після того, як стаття буде створена, відредагуйте її та вставте код, який ми скопіювали (рис. 167). Натискаючи назву статті, ми перейдемо до його режиму редагування, де ми можемо, наприклад, написати опис гри.

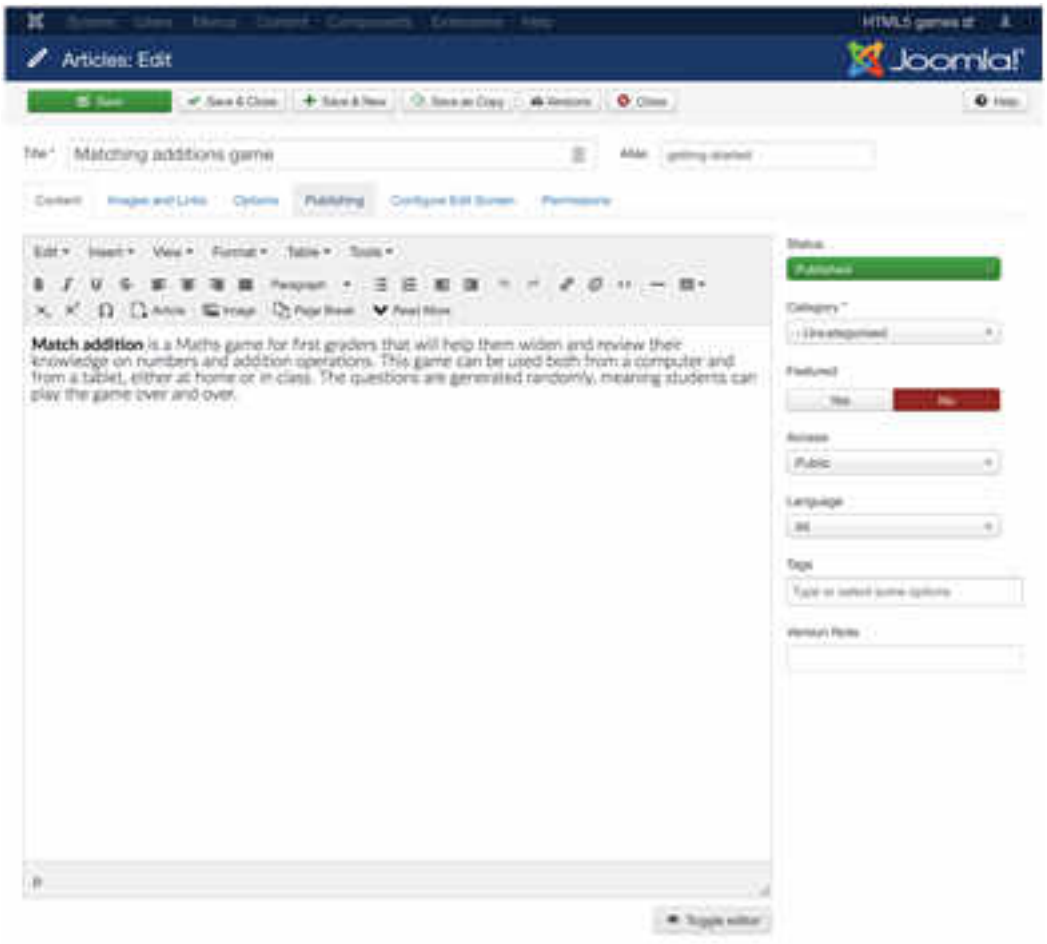

Рисунок 167. Вставка коду

Після опису тексту ми будемо вставляти фрагмент коду, який ми скопіювали з educa.land. Перш ніж вставити код, клацніть правою нижньою кнопкою "Перемкнути редактор" (рис. 168). Після вставлення коду ви побачите, що гру тепер видно після того, як ще раз натиснете кнопку "Змінити редактор".

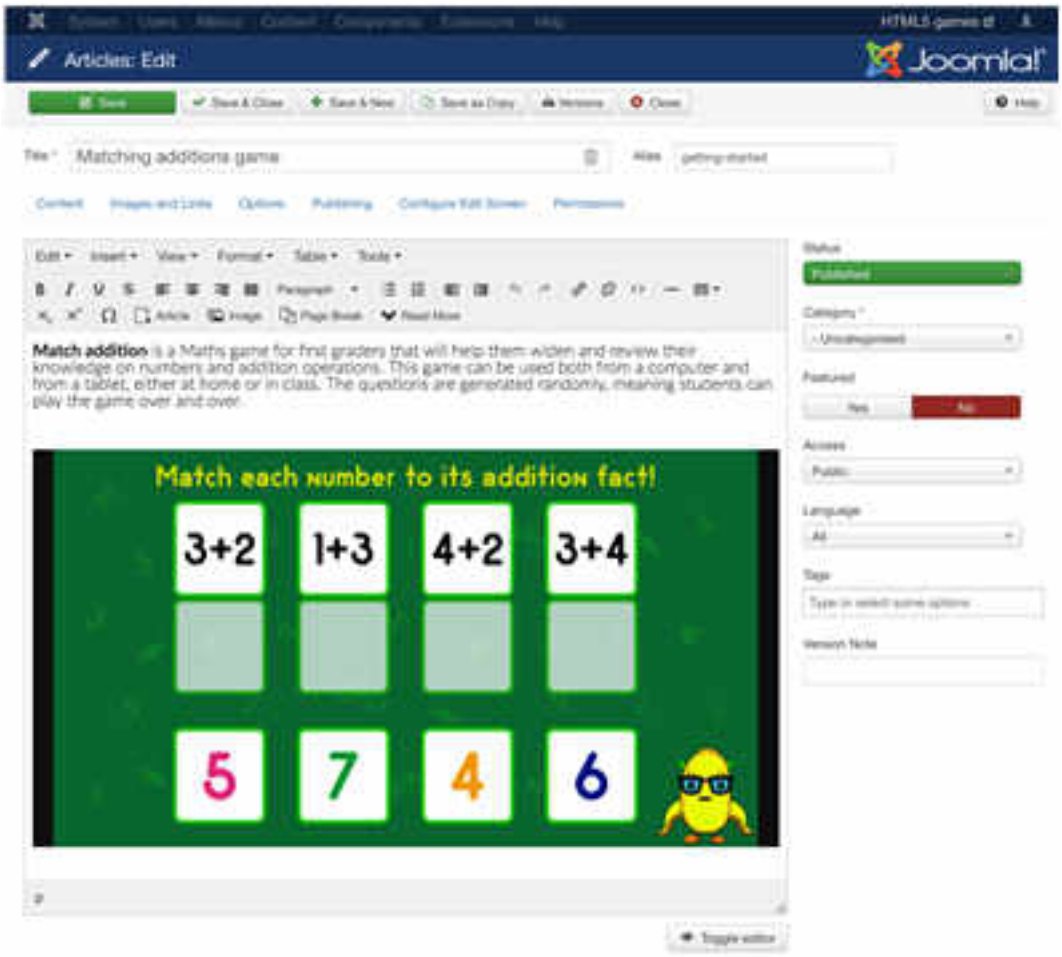

Рисунок 168. Перемикання редактору

Якщо вищенаведене не працює, можливо, це пов'язано з плагіном за замовчуванням для редагування статей, який називається "Редактор - TinyMCE", який налаштований на заборону елементів iframe.

Щоб зупинити це, нам потрібно перейти до налаштувань плагіна (Розширення> Плагіни> Редактор - TinyMCE), а в опції «Заборонені елементи» видалити «iframe» (рис. 169).

І саме так ви отримуєте свою гру на веб-сайті, створеному Joomla (рис. 170).

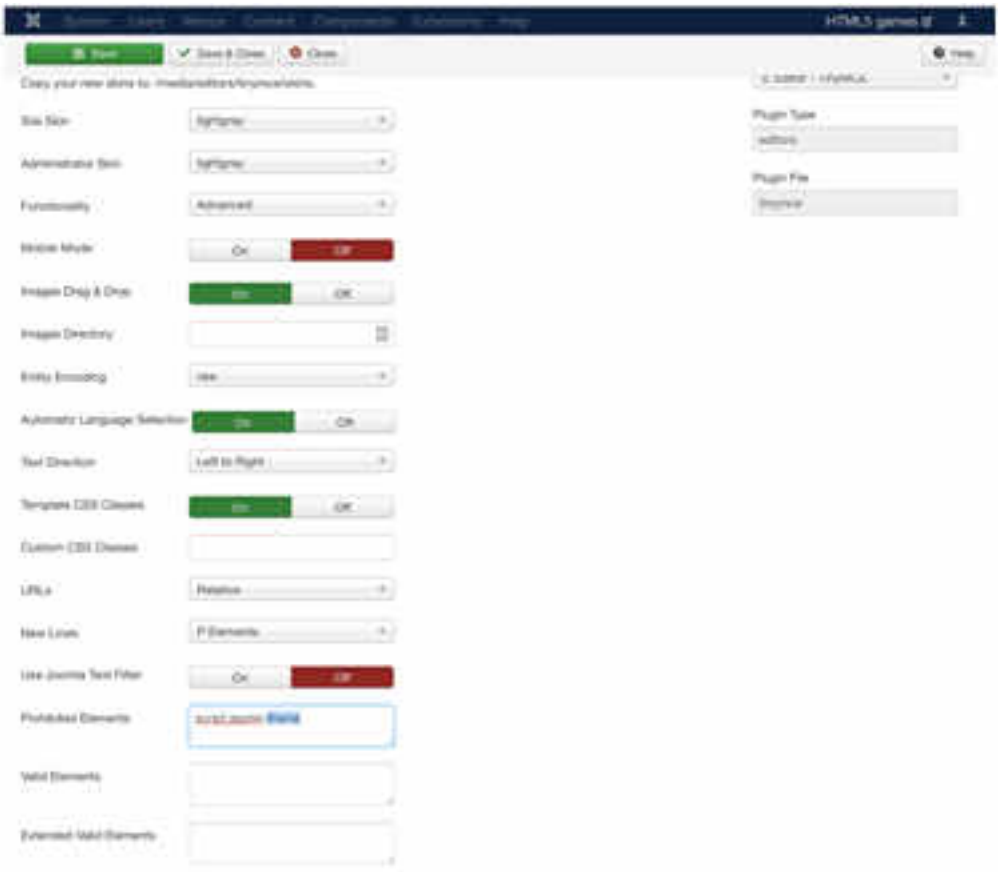

Рисунок 169. Подолання заборони елементів

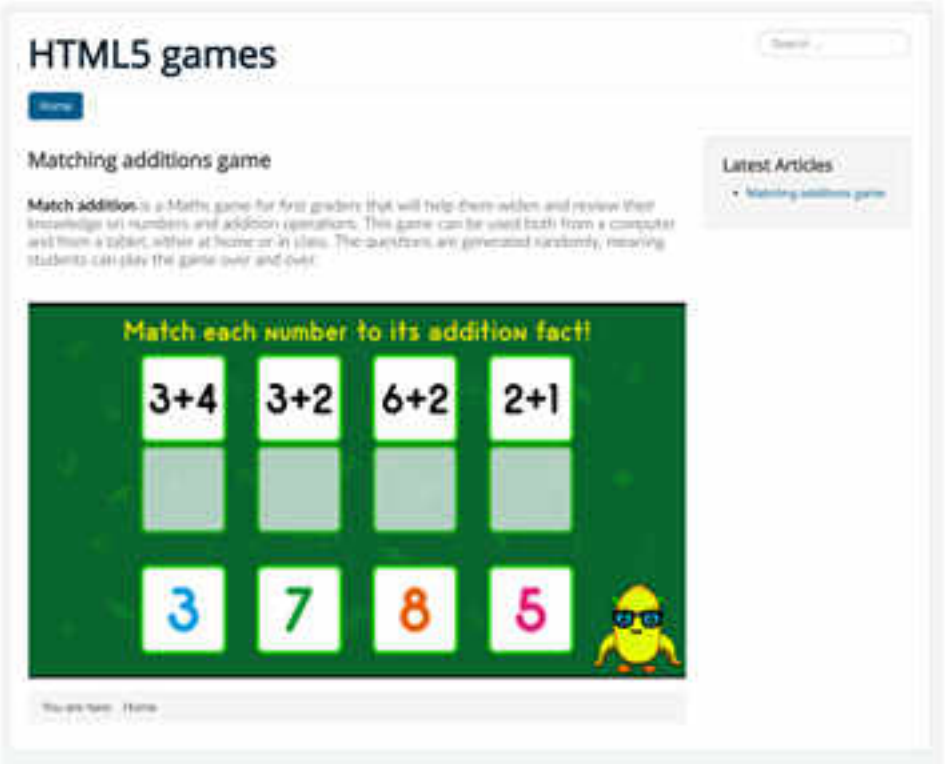

Рисунок 170. Результат додавання гри

ІНТЕГРАЦІЯ ГРИ В BLOGGER

Інтеграція гри HTML5 за допомогою "iframe" в Blogger дуже проста. По-перше, зверніться до адміністратора свого блогу та створіть нову статтю; назвіть її "Гра з доповненнями" (рис. 171).

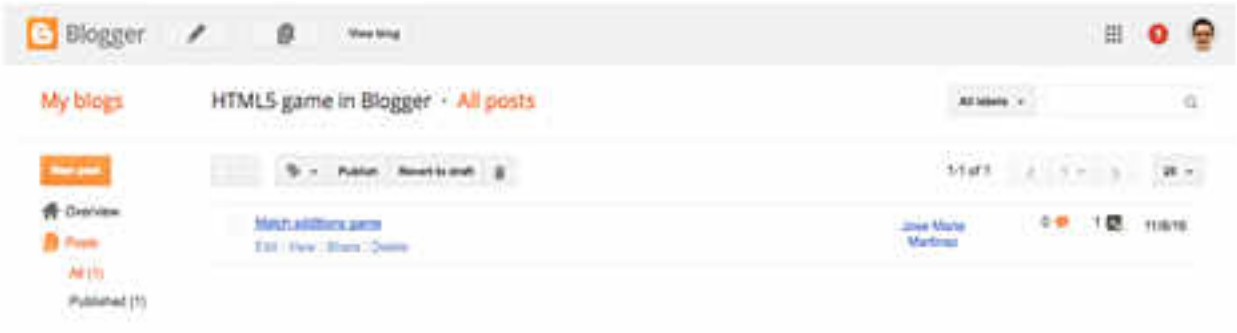

Рисунок 171. Додавання нової статті

Після створення редагуйте його в звичайному режимі, щоб розмістити, наприклад, описовий текст гри (рис. 172).

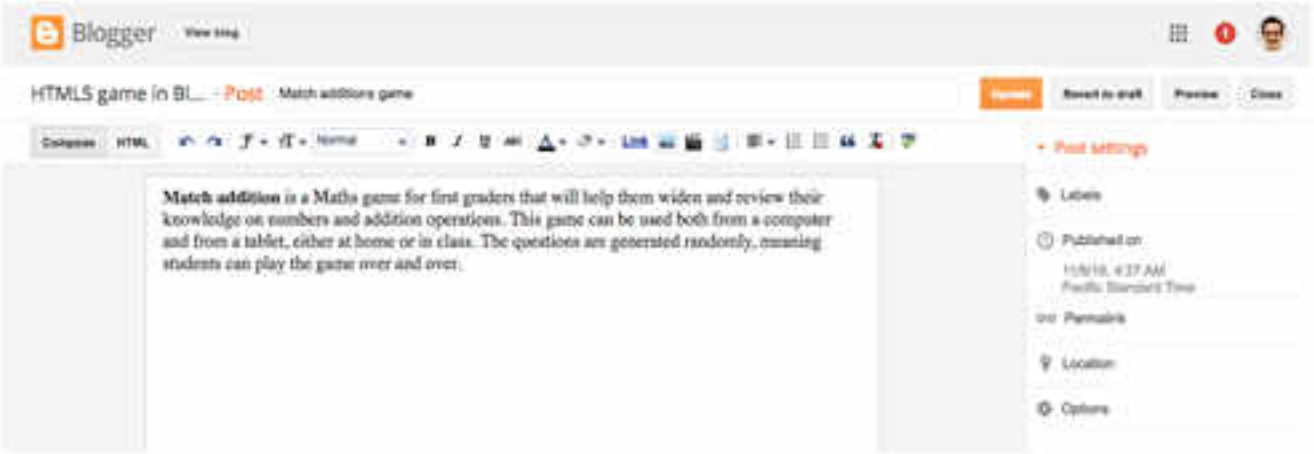

Рисунок 172. Додавання описового тексту гри

Далі натисніть на кнопку HTML та вставте фрагмент коду (рис. 173), який ми скопіювали з гри в educa.land.

Опублікуйте статтю, і ви можете побачити, як гра виглядає на вашому веб-сайті, створеному Blogger (рис. 174).

Рисунок 173. Додавання фрагменту коду

Рисунок 174. Результат додавання гри

ІНТЕГРАЦІЯ ГРИ В MOODLE

Moodle є однією з найпопулярніших і широко використовуваних онлайн-навчальних платформ у світі освіти, яка поширюється безкоштовно як безкоштовне програмне забезпечення відповідно до GNU Public License (GPL).

Це віртуальне навчальне середовище призначене, щоб допомогти педагогам створювати якісні курси в Інтернеті та спрямовані на підтримку конструктивістської системи соціальної освіти.

Moodle - це навчальна платформа, яка, крім можливого використання в дистанційному навчанні, є також важливим інструментом для округлення навчання на місці.

Перш ніж розпочати роботу, якщо ви не знайомі з створенням курсів у Moodle, перегляньте розділ довідки.

Ми збираємося навчитися інтегрувати гру WiMi5 в курс Moodle. Це можна зробити двома способами, як активність або як ресурс.

Ми розглянемо обидва приклади.

ІНТЕГРАЦІЯ ГРИ ЯК РЕСУРСУ В КУРС MOODLE.

Ми почнемо з найпростішого прикладу, інтегруючи навчальну HTML5-гру WiMi5 як ресурс у попередньо створеному курсі Moodle.

Ми знайдемо ресурс всередині першої теми, "Тема 1".

По-перше, зверніться до курсу як адміністратора курсу або творця та увімкніть режим редагування (рис. 175).

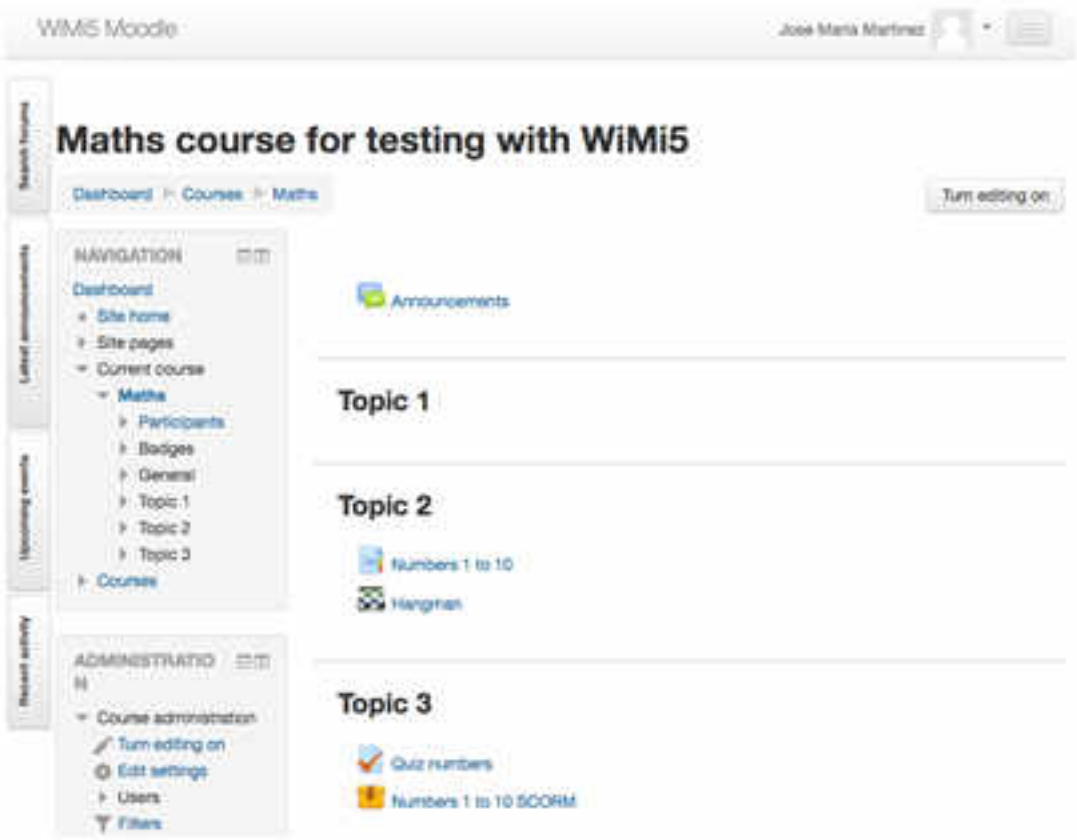

Рисунок 175. Режим редагування курсу

Після ввімкнення режиму редагування з'явиться можливість додавати активність або ресурс у кожній темі (рис. 176).

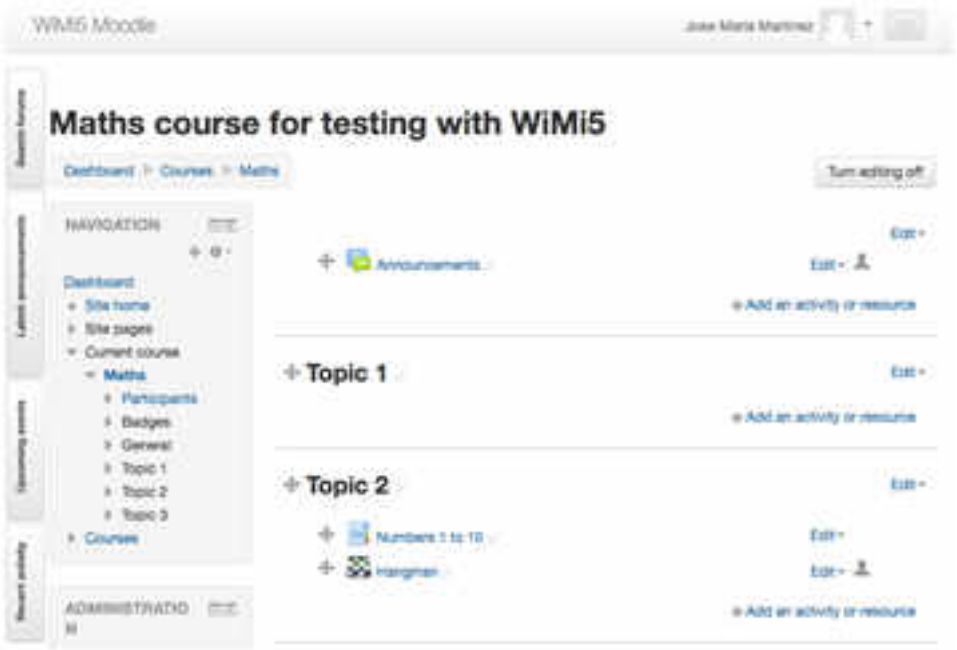

Рисунок 176. Додавання гри як ресурсу

Після натискання кнопки, щоб додати активність або ресурс, з'явиться вікно, де ви зможете вибрати, чи потрібно додати рекламну активність або ресурс, і де ви хочете додати його. У цьому випадку ми збираємося додати URL-адресу (рис. 177).

Рисунок 177. Додавання URL-адреси

Перш ніж додавати інформацію про те, що наступний крок поставить вас, відкрийте нове вікно веб-переглядача на сторінку освітньої гри, яку хочете додати.

У цьому випадку ми виберемо гру "Числа від 1 до 10", створену за допомогою WiMi5, з освітнього порталу educa.land.

На цій сторінці ми будемо копіювати URL-адресу гри, яка з'явиться в розділі "Посилання", натиснувши кнопку "Скопіювати в буфер обміну" (рис. 178).

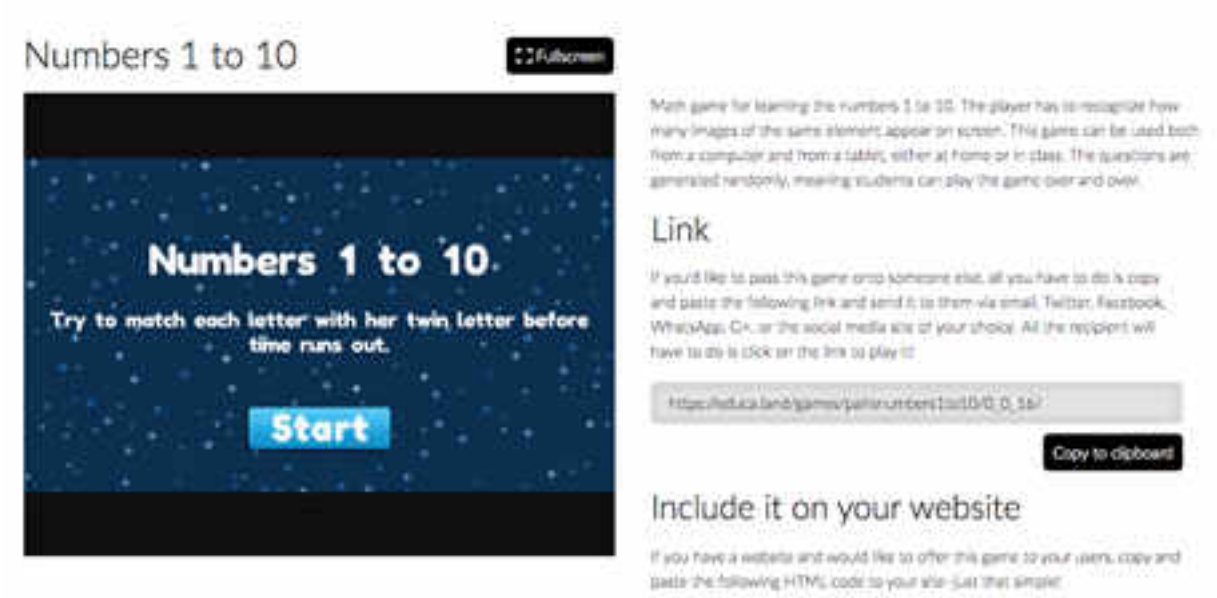

Рисунок 178. Копіювання URL-адреси гри

Тепер поверніться до адміністратора Moodle та на екрані, де ви додаєте URL-адресу, заповніть дані, які він запитує для нашого курсу (рис. 179).

У цьому випадку заповніть заголовок ресурсу, його URL-адресу (яку ми просто скопіювали з educa.land) та опис гри, який ми можемо знайти на сторінці курсів, якщо ми перевіримо відповідне поле.

Після збереження введеної інформації ви зможете побачити, як ваша гра з'являється в рамках курсу Moodle (рис. 180).

ІНТЕГРАЦІЯ ГРИ ЯК АКТИВНОСТІ У КУРСІ MOODLE

У Moodle для створення курсів можуть бути використані як ресурси, так і як активність. Зазвичай ресурси - це елементи, які дозволяють студентам отримувати доступ до вмісту.

З іншого боку, активність є робочим інструментом для студентів. Активність зазвичай має певну оцінку, будь то автоматична або виправлена репетитором.

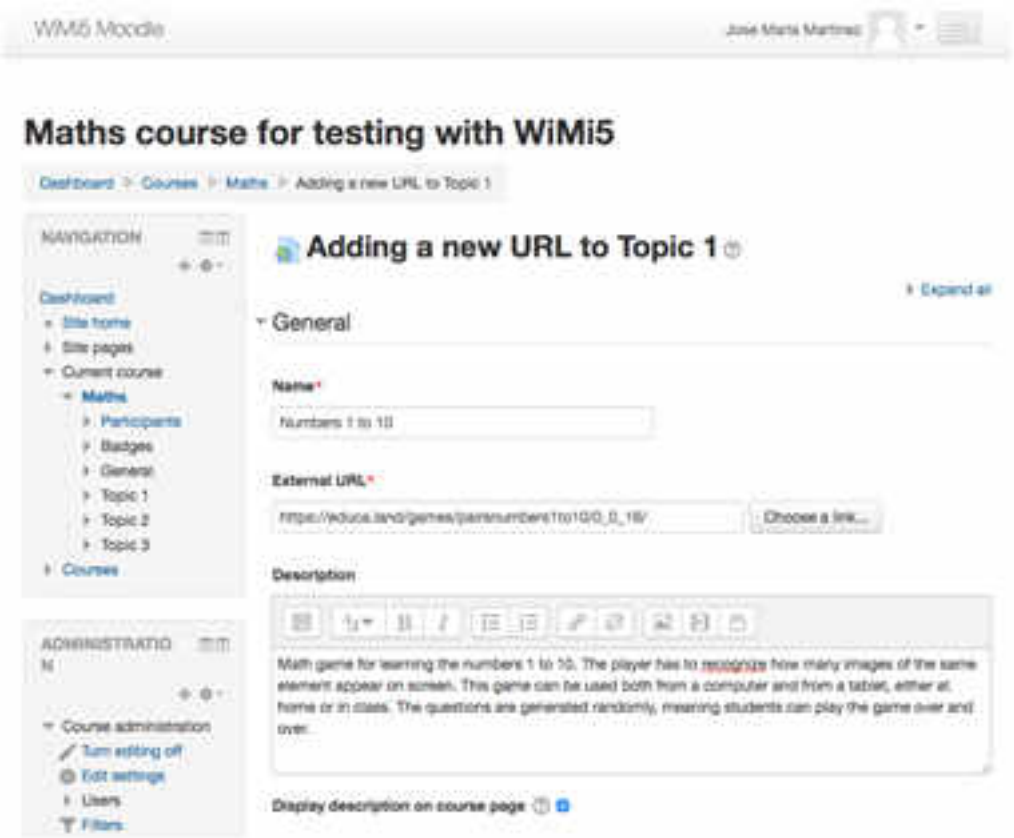

Рисунок 179. Розміщення даних

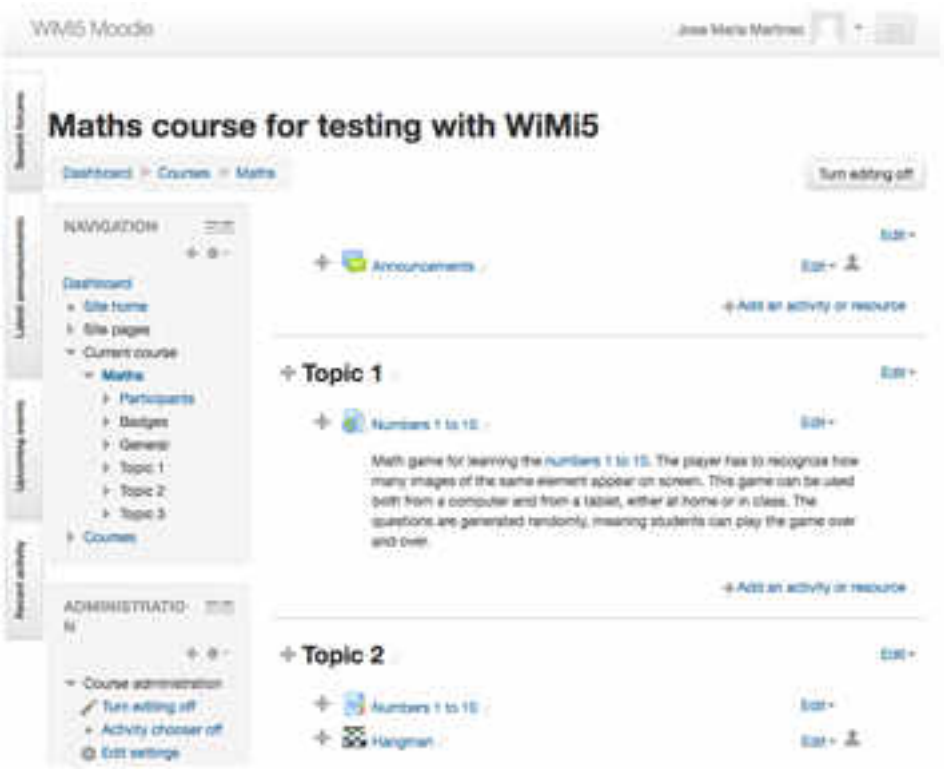

Рисунок 180. Поява гри як ресурсу курсу

Щоб додати активність, ще раз натисніть на кнопку додавання ресурсів або активності та виберіть параметр "SCORM package" (рис. 181).

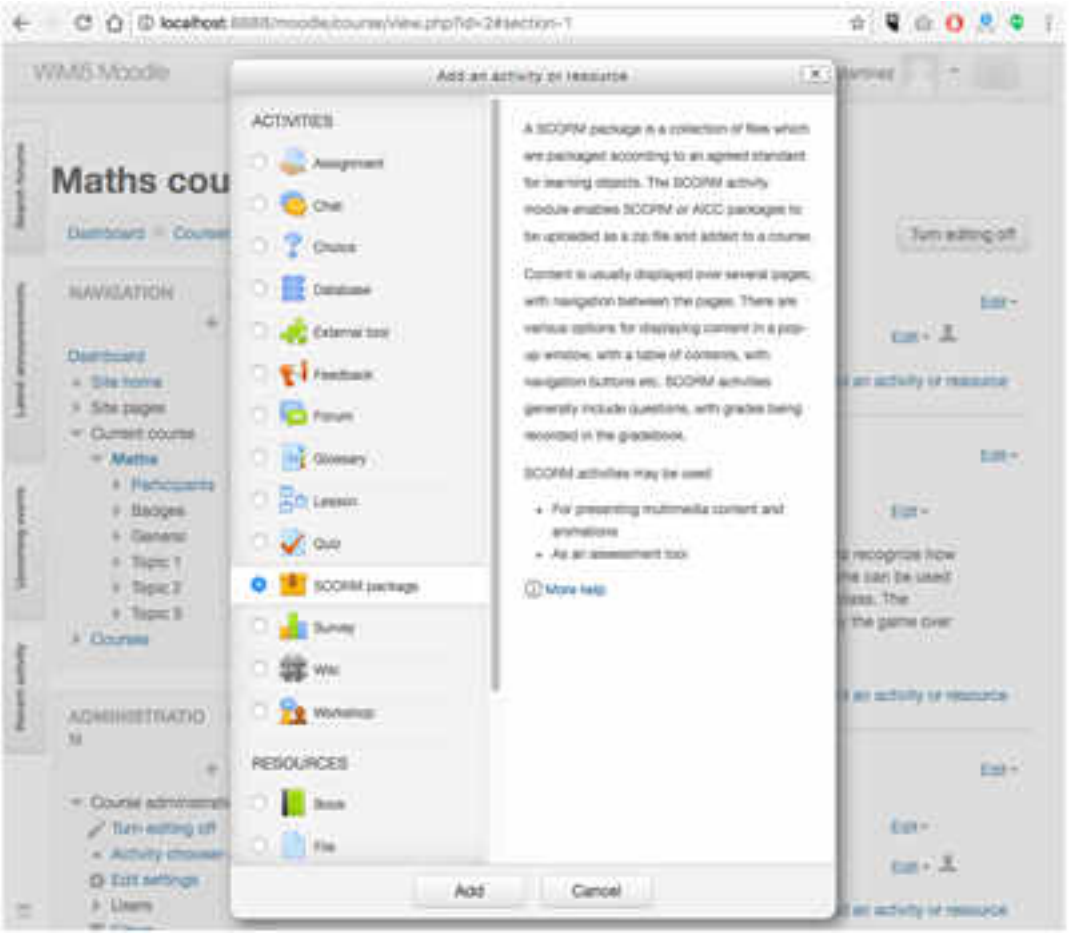

Рисунок 181. Додавання гри як активності

Для створення пакета SCORM ми збираємося використовувати безкоштовний інструмент із відкритим кодом під назвою eXeLearning.

Цей інструмент дозволяє нам експортувати файли в будь-який з декількох форматів використання навчальних платформ. Ми будемо використовувати формат SCORM 2004, який є поточною версією SCORM. SCORM (Sharable Object Reference Model) - це набір стандартів та специфікацій, який дозволяє створювати структуровані педагогічні об'єкти.

Завантаживши та встановивши eXeLearning, запустіть програму, щоб ви могли створити пакет SCORM.

Виберіть параметр "Зовнішній веб-сайт" на вкладці "Нетекстові відомості" та введіть інформацію, яку запитує програма (рис. 182).

Після його збереження ви можете бачити завантаження вашої гри, а потім все, що вам потрібно зробити, це експорт вмісту у форматі SCORM 2004.

Для цього виберіть Файл> Експорт> Стандарт освіти> SCORM 2004 (рис. 183).

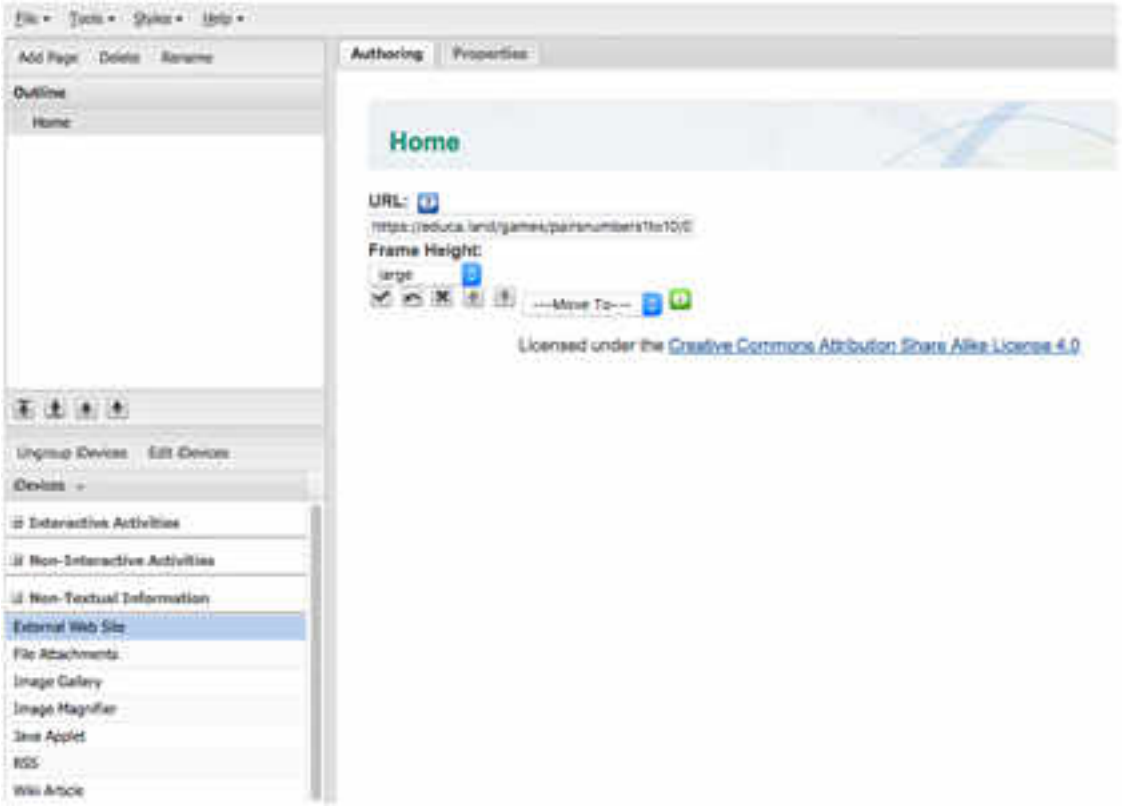

Рисунок 182. Гра Пошук серії слів

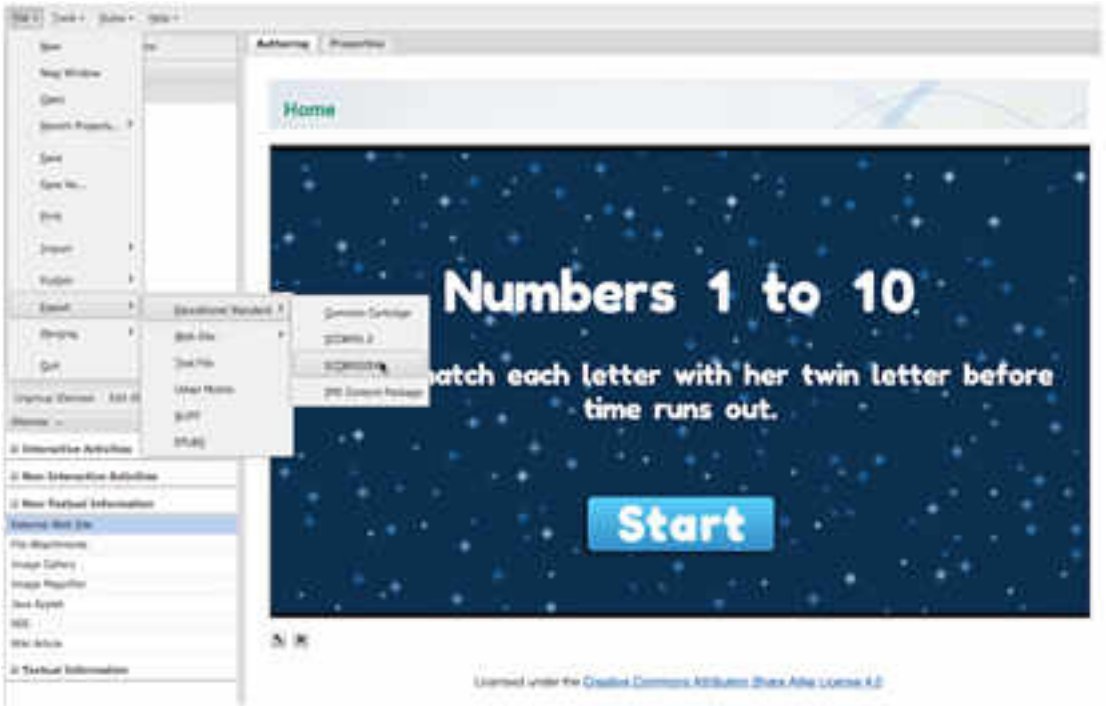

Рисунок 183. Експорт пакету гри

Тепер, коли файл збережений, все, що вам потрібно зробити, це додати його як пакет SCORM 2004 у вашому курсі Moodle.

Поверніться до вікна браузера, де ви додали пакет SCORM та заповнили дані. Вам потрібно буде завантажити ваш SCORM-пакет, збережений як файл .zip, у розділі "Пакет" (рис. 184).

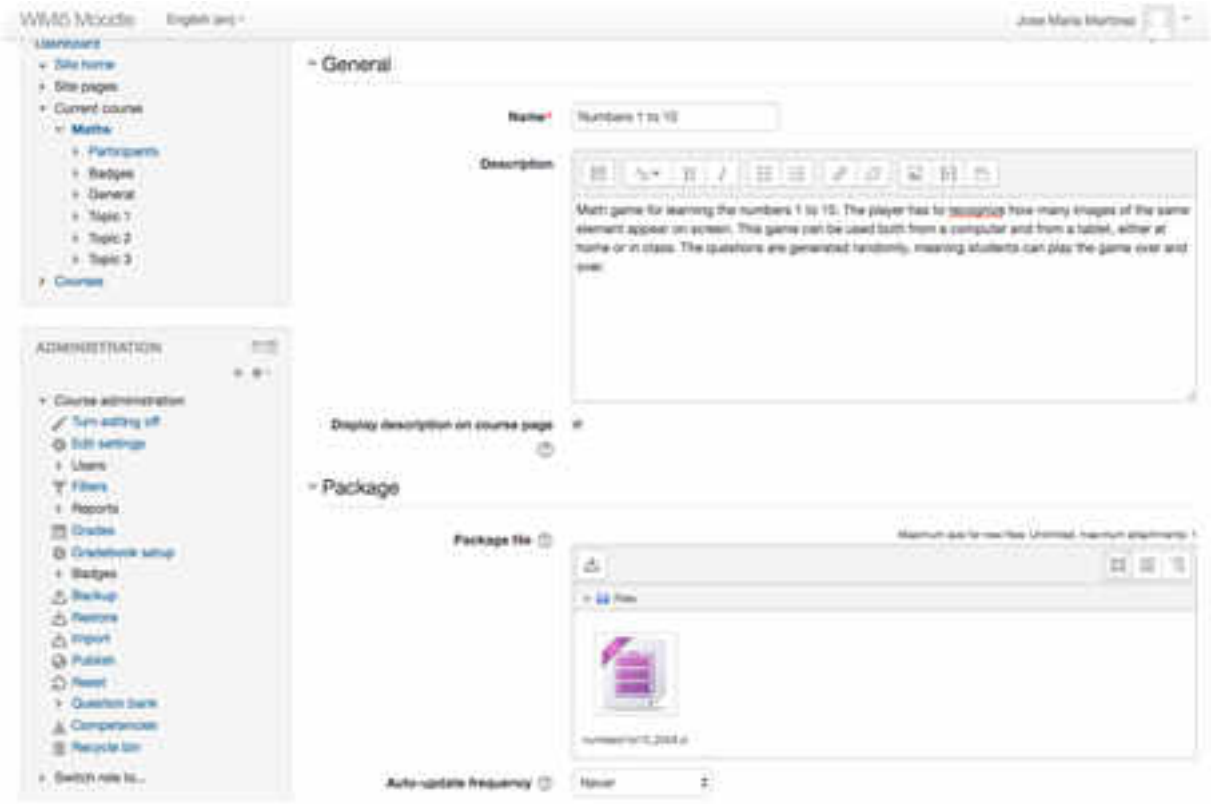

Рисунок 184. Завантаження пакету гри

Як тільки ви це зробите, ви побачите, як зараз курс містить навчальну гру WiMi5 як активність та ресурс (рис. 185).

Все, що залишилося, полягає в тому, щоб вимкнути режим редагування та спробувати вміст (рис. 186).

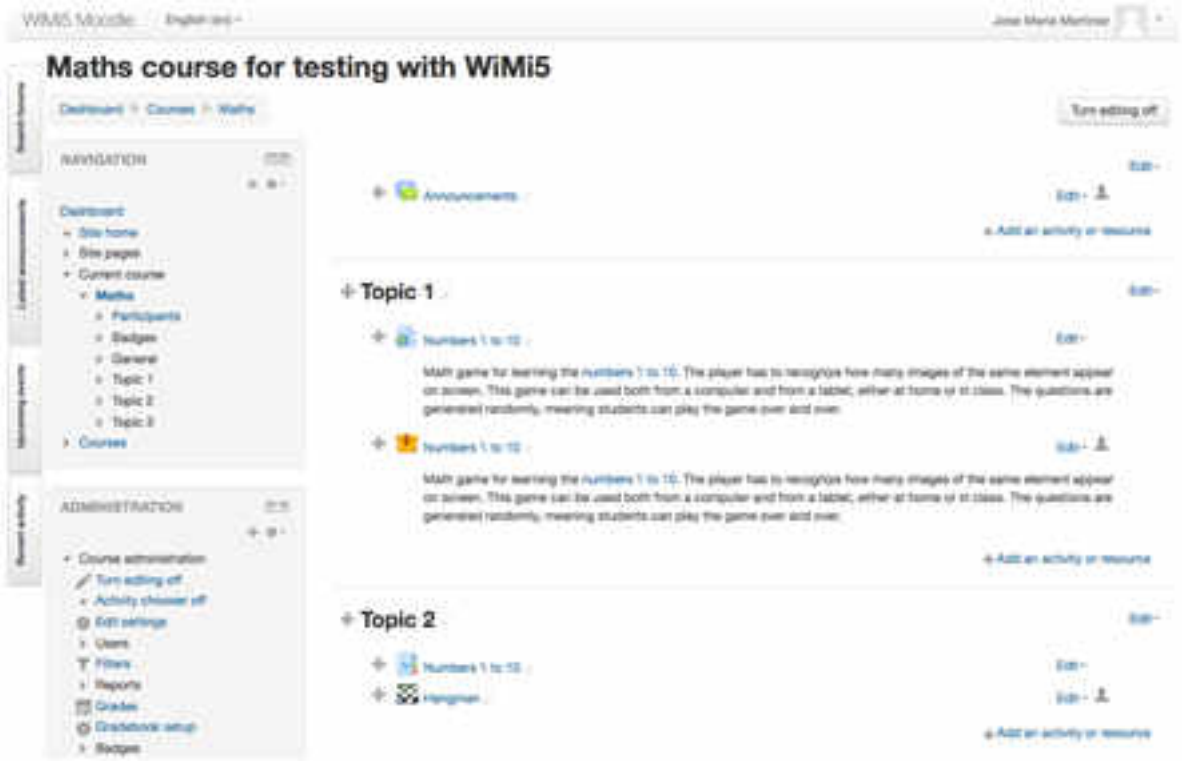

Рисунок 185. Гра як активність та ресурс курсу

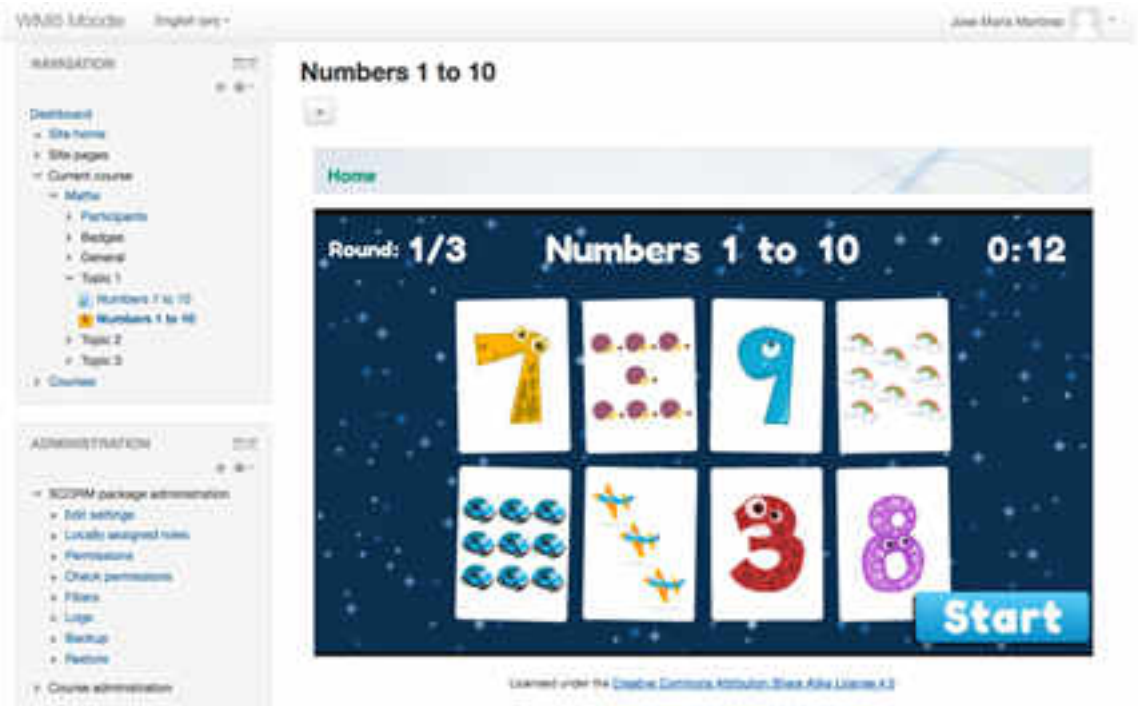

Рисунок 186. Режим перегляду ресурсів

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

- Що є комп'ютерною грою?
- Що таке ігровий процес?
- Які є види гравців?
- Як враховуються особливості різних типів гравців при розробці гри?
- Які різновиди комп'ютерних ігор Ви знаєте?
- Які є сфери застосування ігор різних типів?
- Які етапи розвитку комп'ютерних ігор Ви знаєте?
- Які етапи розробки комп'ютерної гри Ви знаєте?
- Який склад має команда розробників гри?
- Які ролі відіграють члени команди розробників в процесі розробки гри?
- Які вимоги до професійних компетенцій кожного з учасників команди розробників гри?
- Що таке ігровий жанр?
- Що таке візуалізація гри?
- Що таке ігровий рушій?
- Що таке ігровий конструктор?
- Що таке життєвий цикл комп'ютерної гри?
- З яких етапів складається життєвий цикл комп'ютерної гри?
- В чому полягають задачі кожного з етапів життєвого циклу комп'ютерної гри?
- В чому полягає взаємозв'язок етапів життєвого циклу комп'ютерної гри?
- Яку структуру має ігровий додаток?
- В чому полягають особливості архітектури ігрових додатків?
- Які існують методи проектування та розробки ігрових додатків?
- В чому полягають особливості командної розробки КІД?
- Які існують види інструментальних засобів розробки ігрових додатків?
- Якими є сфери застосування різних типів ігрових рушіїв?
- Що таке геймплей?
- Що таке геймабіліті?
- Що таке ігрова модель та з чого вона складається?
- Що таке сюжет гри?
- Що таке стан гри?

- Що таке сцена гри?
- Як описати правила гри?
- Як подаються сценарії гри?
- Що таке ігровий баланс?
- Як визначити множину шляхів проходження гри?
- Як забезпечити ігровий баланс у комп'ютерній грі?
- Що таке крива розвитку гравця та як вона залежить від сценарію?
- Що таке Storytelling?
- Що таке Storyboard?
- З чого складається ігрове середовище?
- Від чого залежить динаміка ігрового середовища?
- Що таке ігрова роль?
- Що таке персонаж?
- В чому полягає динаміка персонажу?
- Які ігрові обєкти належать до ігрового середовища?
- Що таке ігровий ресурс?
- Які є форми існування ігрового ресурсу?
- Як створюються персонажі та об'єкти у комп'ютерній грі?
- Як добирається дизайн персонажів в залежності від їх ролей в комп'ютерній грі?
- Яким є вплив візуальних та звукових ефектів на сприйняття гравцем ігрового середовища?
- Які інструментальні засоби використовуються для створення ігрових персонажів, об'єктів та ресурсів?
- В чому полягає взаємодія між ігровими ресурсами, об'єктами та персонажами ігрового середовища?
- В чому полягає сутність ідеї комп'ютерної гри?
- З чого складається концепція комп'ютерної гри?
- Як необхідно окреслювати та втілювати ідею комп'ютерної гри?
- В чому полягає послідовність створення концепції комп'ютерної гри?
- Як окреслити переваги запропонованої концепції комп'ютерної гри?
- Як знайти коло потенційних гравців?
- Як обґрунтувати актуальності комп'ютерної гри?
- Як необхідно презентувати концепцію комп'ютерної гри для пошуку інвестора?

- Що таке ігрове середовище?
- Що є персонажем гри?
- Що є ресурсом гри?
- Що є об'єктом гри?
- За якими принципами виконується дизайн персонажів?
- Які інструменти використовують для створення ігрових ресурсів?
- Які ефекти можуть бути використані для подання ігрових ресурсів?
- Як забезпечується динаміка гри?
- Які різновиди графіки використовуються в комп'ютерних іграх та для вирішення яких задач?
- Яким є взаємозв'язок ігрової моделі та ігрового середовища?
- Охарактеризуйте середовище візуальної розробки WIMI5.
- Поясніть функціональні можливості середовища WIMI5.
- Охарактеризуйте можливості роботи з ігровими об'єктами та ресурсами.
- Поясніть призначення вікон інтерфейсу середовища WIMI5.
- Поясніть призначення елементів інтерфейсу користувача середовища WIMI5.
- Поясніть структуру ігрового проекту, що створюється середовищем WIMI5.
- Назвіть сферу застосування середовища WIMI5.
- З чого складається середовище WIMI5?
- Що таке редактор логіки?
- Якими є можливості редактора логіки?
- Що таке редактор сцен?
- Якими є можливості редактора сцен?
- Що таке чорна скринька?
- Що таке сцена?
- Як налаштувати логіку гри?
- Що таке шаблон гри?
- В чому полягає налаштування шаблону гри?
- В чому полягає адаптація шаблону гри?
- Як створюється проект гри «з нуля»?
- Як розробити візуалізацію гри?
- Що таке ігровий ресурс?
- Що таке спрайт?

- Як організовані ігрові ресурси на робочому столі?
- З яких елементів складається логіки гри?
- Для чого призначений менеджер ресурсів?
- Що таке сцена?
- Як організовані переходи між сценами?
- Що таке менеджер ресурсів?
- Якими є елементи логічної схеми?
- Що таке компоненти з'єднання?
- Які панелі має логічна схема?
- Якими є основні типи чорних скриньок?
- Що таке логічне дерево?
- Для чого призначений менеджер активів?
- Що таке конектор і як він використовується?
- Що таке тригер та в яких випадках він активується?
- Що таке властивість та в яких об'єктах вона присутня?
- Що таке параметр та які його види бувають?
- Які види вихідних конекторів Ви знаєте?
- Що таке колекція параметрів та як отримати доступ до її елементів?
- Що таке посилання та як воно реалізується?
- Які елементи реалізуються при завантаженні гри? Рівня?
- Що таке потік логіки гри?
- Які є типи чорних скриньок?
- Як реалізуються посилання на властивість параметра?
- Що можна налаштувати в елементі CodeRunner?
- Як інтегрувати JavaScript код у ігровий додаток?
- Що таке віртуальні товари?
- Як створити віртуальні товари?
- Як налаштувати конфігурацію віртуальних товарів?
- Що таке віртуальна чорна скринька?
- Як організовано віртуальний магазин?
- За допомогою якого інструменту публікують ігрові проекти?
- Як протестувати ігровий проект?
- Як розгорнути ігровий додаток?
- Як публікують ігровий додаток?
- Як використовувати сайт WIMI Labs?
- Як створити завантажуваний файл проекту?

- Як опублікувати гру HTML5 в інтернет-магазині Google Chrome?
- Поясніть, які інструменти і як саме можуть бути використані для візуальної розробки елементів ігрового середовища.
- Поясніть функціональні можливості середовища WIMI5 щодо реалізації графічних, анімаційних та звукових ігрових об'єктів.
- Охарактеризуйте можливості роботи з ігровими об'єктами та ресурсами.
- Поясніть способи розв'язання задач взаємозв'язку ігрових об'єктів у WIMI5.
- Поясніть, які інструменти і як саме можуть бути використані для візуальної розробки елементів ігрової моделі.
- Поясніть функціональні можливості середовища WIMI5 щодо реалізації сценаріїв взаємодії ігрових об'єктів.
- Охарактеризуйте можливості реалізації руху та зіткнень ігрових об'єктів.
- Поясніть способи розв'язання задач побудови потоку логіки гри засобами WIMI5.
- Поясніть, які інструменти монетизації і як саме можуть бути використані для ігрових додатків.
- Поясніть, які інструменти розгортання і як саме можуть бути використані для ігрових додатків.
- Поясніть, які інструменти публікації і як саме можуть бути використані для ігрових додатків.
- Поясніть функціональні можливості середовища WIMI5 щодо монетизації, розгортання та публікації комп'ютерних ігор.
- Охарактеризуйте можливості розповсюдження розроблених ігрових додатків за допомогою WIMI5.

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ ГРУПОВОГО ФІНАЛЬНОГО ПРОЕКТУ

1. Розробити гру "Намети".

Правила гри такі. Таблиця 10х10 є план лісу, в якому зазначено положення 20 дерев. До кожного дерева потрібно прив'язати свій намет (в клітку, що торкається клітини дерева стороною), всього рівно 20 наметів. Клітини з наметами не повинні торкатися один одного (навіть кутом!). Числа зліва і вгорі позначають кількість клітин, зайнятих наметами, у відповідному рядку або стовпці.

Вимоги до програми:

-програма повинна бути виконана мовою JavaScript;

-в програмі передбачити розділ з правилами гри;

-ігрок (користувач) повинен переміщати фішки за допомогою маніпулятора «миша»;

-в грі передбачити лічильник ходів і скасування останнього зробленого ходу.

2. Розробити ігрову програму «Хрестики-нулики», яка включає в себе ігрову панель, що складається з дев'яти клітин (панелей), що утворюють собою один великий квадрат. Ігрова програма розрахована на двох гравців. Супротивникам по черзі надається хід, один грає символом «Х», інший - «О». Виграє той, хто першим заповнить ряд панелей своїм символом: будь-який з трьох рядів по горизонталі або вертикалі, або будь-який ряд по діагоналі.

3. Розробити гру "Бики і корови". Програма повинна дозволяти грати двом гравцям і гравцеві проти комп'ютера. Кожен з супротивників задумує чотиризначне число, всі цифри якого різні (перша цифра числа відмінна від нуля).

Необхідно розгадати задумане число. Виграє той, хто відгадає перший. Противники по черзі називають один одному числа і повідомляють про кількість "биків" і "корів" в названому числі ("бик" - цифра є в запису задуманого числа і знаходиться в той же позиції, що і в задуманому числі; "корова" - цифра є в запису задуманого числа, але знаходиться не в той же позиції, що і в задуманому числі).

Наприклад, якщо задумано число 3275 і названо число 1234, отримуємо в названому числі одного "бика" і одну "корову". Очевидно, що число відгадане в тому випадку, якщо маємо 4 "бика".

4. Розробити гру "Тетріс".

Правила гри такі. Випадкові фігурки тетраміно падають зверху в прямокутний стакан шириною 18 і висотою 19 клітин. У польоті гравець може повертати фігурку на 90 ° і рухати її по горизонталі. Також можна «скидати»

фігурку, тобто прискорювати її падіння, коли вже вирішено, куди фігурка повинна впасти. Фігурка летить до тих пір, поки не наткнеться на іншу фігурку або на дно склянки. Якщо при цьому заповнився горизонтальний ряд, він пропадає, а все що вище нього, опускається на одну клітку. Додатково показується фігурка, яка буде слідувати після поточної - це підказка, яка дозволяє планувати гравцеві дії. Темп гри поступово збільшується (-30мсек). Гра закінчується, коли нова фігурка не може поміститися в стакан. Гравець отримує очки за кожен заповнений ряд, тому його завдання - заповнювати ряди, не заповнюючи сам стакан (по вертикалі) якомога довше, щоб таким чином отримати якомога більше очок.

Нарахування очків в різних версіях «Тетріс» досить різноманітне. Очки можуть нараховуватися за прибрані лінії, за скинуті фігурки, за перехід на нову швидкість тощо.

При нарахуванні очків за лінії кількість очків зазвичай залежить від того, скільки ліній прибрано за один раз. Наприклад, в китайських «Тетріс», популярних в СНД в 1990-х роках, нарахування очків зазвичай було таким:

1 лінія - 10 очок;

2 лінії - 30 очок;

3 лінії - 60 очок;

4 лінії (тобто, зробити Тетріс) - 100 очок.

Тобто, чим більше ліній забирається за один раз, тим більше відношення кількості очок до кількості ліній. Цікаво, що тетрісом в багатьох версіях гри також називається дію, після якого зникає відразу 4 лінії. Це можна зробити тільки одним способом - скинути «палицю» (фігурку, в якій всі клітини розташовані на одній лінії) в «шахту» ширини 1 і глибини як мінімум 4.

5. Розробити гру "Куточки".

Правила гри такі. На дошці 10 х 10 розставлені фішки двох гравців квадратами 4 х 4 так, що вони займають протилежні кути дошки. Гравці ходять по прямій (по черзі) і можуть пересувати свою фішку на сусідню (вільний) поле, або перестрибувати через сусідню фішку, якщо за нею є вільне поле. Дозволяється багатоходові стрибки (як в шашках), якщо є така можливість. Перемагає той, хто першим перебудує свої фішки, повністю перевівши їх в кут супротивника.

5. Розробити гру «Сапер».

Правила гри такі. Сапер являє собою логічну гру, основною метою якої є знаходження всіх захованих бомб на мінному полі. Ваше завдання відкрити всі осередки поля, що не містять бомб, заблокувавши (позначивши) при цьому осередку, в яких розташовані бомби. Поле гри задано у вигляді двомірного масиву. У цьому масиві розташовані осередки. Спочатку вони все закриті. Осередки можуть бути порожні, з цифрами і з бомбами. Для кожної такої комірки ми задаємо клас. При натисненні лівої кнопки миші відкривається

осередок, при натисканні правої кнопки, виставляється прапорець, при цьому лівою кнопкою миші ви вже не можете натиснути на заблоковану осередок, але правою кнопкою можна зняти позначку прапорця. Осередок з цифрою позначає те, скільки хв знаходиться в окрузі цього осередку. При першому натисканні кнопки миші на осередок не може відкритися осередок з бомбою.

Гра вважається програною, якщо ви відкрили осередок з бомбою. Гра вважається виграною, якщо на ігровому полі все осередки з бомбами відзначені і всі інші осередки - відкриті.

6. Розробити гру "Лабіринт".

Метою цієї гри є знаходження виходу з лабіринту. Персонажем, для руху по лабіринту, є червоний квадрат, рух здійснюється за допомогою стрілок клавіатури - вгору, вниз, вправо, вліво. У грі дається час, обмежене однією хвилиною, якщо користувач не пройде до виходу за хвилину, то відповідно гра для нього закінчена і йому надається можливість пройти лабіринт заново.

7. Розробити гру «Морський бій».

Правила гри такі. На ігровому майданчику розміром 10 на 10 клітин Гравець розставляє один корабель розміром чотири клітини, два корабля розміром три клітини, три корабля розміром дві клітини і чотири кораблі розміром в одну клітку (для полегшення кораблі розставляються автоматично). При цьому корабель представляє собою послідовність сусідніх клітин, що стоять на одній вертикалі або на одній горизонталі. Сусідні кораблі не повинні мати спільних точок. Противником Гравця є Комп (Комп'ютер), який автоматично розставляє кораблі на своєму полі за вказаними вище правилами.

Після розстановки починається бій. Він являє собою почергові постріли Гравця і Компа. При попаданні в корабель противника учасник бою отримує можливість проведення позачергового пострілу. Гра закінчується при знищенні одним із учасників усіх кораблів противника.

8. Розробити гру "Змійка".

Опис ігрового процесу. Ігрове поле являє собою прямокутне вікно розміром 640x480 пікселів. На полі з'являється «Змійка» - квадрат чорного кольору розміром 10x10 пікселів.

Для того, щоб привести змійку в рух потрібно натиснути одну з клавіш. Клавіші: вгору, вниз, вліво, вправо.

Також на полі з'являється яблуко - це квадрат розміром 10x10 пікселів. Місце розташування яблуко визначається випадковим чином, за допомогою класу Random.

Яблуко може бути зеленого кольору (звичайне яблуко при з'їданні якого нараховується окуляри, кількість яких дорівнює номеру рівня), або червоне

яблуко (бонусне, при з'їданні цього яблука нараховуються очки, кількість яких дорівнює номеру рівня помноженому на три).

Коли змійка з'їдає будь-яблуко, незалежно від кольору, її розмір збільшується на один квадрат розміром 10x10 пікселів.

Якщо змійка врізається сама в себе або в одну з чотирьох стін, вона гине і гра закінчується.

Мета гри набрати 1000 очок.

9. Розробити гру, в уоторой здійснюється управління гравцем (Player), який повинен зловити наживку (Bait) - і при цьому ухилитися від з'являються "хижаків" (Barrier). Мета гри - зловити максимальну кількість наживок, що не зіткнувшись з хижаками. При зіткненні з одним з хижаків все вони пропадають, а окуляри - обнуляються, що фактично, рівносильно початку гри з нуля.

10. Розробити гру з наступними правилами. Чорний прямокутник представляє гравця, чиє завдання - збирати жовті квадрати (монети), уникаючи червоних ділянок (лава?). Рівень закінчується, коли гравець зібрав всі монети.

Гравець може ходити клавішами вліво і вправо, і стрибати клавішею вгору. Стрибки - це спеціальність нашого персонажа. Він може стрибати в кілька разів вище свого росту і міняти напрям руху в повітрі. Це не дуже-то реалістично, але допомагає гравцеві відчути повний контроль над його екранним аватаром.

У гри фіксований фон у вигляді решітки, де рухомі елементи накладаються на фон. Кожна осередок решітки або порожня, або є стіною, або лавою. Рухомі елементи - гравець, монети, і деякі шматочки лави. На відміну від симуляції життя з глави 7, позиції цих елементів не прив'язані до грат. Їх координати можуть бути дробовими, забезпечуючи плавний рух.

11. Завдання гри - виростити свою колонію жуків і знищити всіх юнітів супротивника. Жуки хаотично бігають по ігровому полю, а завдання гравця - допомагати їм знаходити бонуси у вигляді різної їжі і надавати допомогу своїм юнитам, на яких було вироблено напад.

Бонуси в грі:

кекс - дає 5xp, при наборі 15xp жук розмножується

яблуко - відновлює 50hp, якщо жук повністю здоровий то додає 15 додаткових hp

перець - збільшує атаку на 5dm

жолудь - дає 2xp і кидається в найближчого супротивника, при попаданні завдає потрійної шкоди

мухомор - дає 1xp і дозволяє зробити отруйний постріл, при попаданні завдає 1/2 шкоди і уповільнює жертву

Грати можуть від 2 до 4 осіб. Можна також просто підключитися до сервера з різних вкладок браузера, і пограти одному.

12. Під землею зберігаються незліченні скарби. Сідаємо за кермо землерийної машини і збираємо все, що погано лежить. Але золото і діаманти охороняють злісні ноббіни і хоббіни. Ноббіни пересуваються тільки по тунелях і з часом перетворюються в хоббінов, які можуть дістатися до Діггера найкоротшим шляхом. Їх можна придавити мішком золота або застрелити з гармати. Гармата перезаряджається довго, тому доведеться маневрувати, різко змінюючи напрямок руху і створювати хитрі системи тунелів, збиваючи противника з пантелику.

Коли все ноббіни вилізуть з лігва, в правому верхньому куті з'являться вишеньки. З'їжте їх, і ви помінялися місцями з монстрами, тепер вони будуть тікати від вас. Але будьте обережні, це триває недовго.

13. Розробити гру з наступними правилами. У грі жаба повинна ловити падаючі квіточки, якщо квіточку впаде на землю, то віднімається "Здоров'я". Жаба повинна уникати зіткнення з комарами. На першому рівні три квіточки і три комара, на кожному наступному рівні комарів і квіточок на 1 більше. Всього в грі 4 рівня.

14. Розробити просту симуляцію життя на прямокутної решітці, кожен елемент якої живої чи ні. У кожному поколінні (кроці гри) застосовуються такі правила:

- кожна жива клітина, кількість сусідів якої менше двох або більше трьох, гине
- кожна жива клітина, у якій від двох до трьох сусідів, живе до наступного ходу
- кожна мертва клітина, яка має рівно три сусіда, оживає

Сусіди клітини - це все сусідні з нею клітини по горизонталі, вертикалі і діагоналі, всього 8 штук.

Зверніть увагу, що правила застосовуються до всієї решітці одночасно, а не до кожної з клітин по черзі. Тобто, підрахунок кількості сусідів відбувається в один момент перед наступним кроком, і зміни, що відбуваються на сусідніх клітках, не впливають на новий стан клітини.

Реалізуйте гру, використовуючи будь-які відповідні структури. Використовуйте Math.random для створення випадкових початкових популяцій. Виводите поле як грати з галочок з кнопкою «перейти на наступний крок». Коли користувач включає або вимикає галочки, ці зміни потрібно враховувати при підрахунку наступного покоління.

КРИТЕРІЇ ОЦІНЮВАННЯ

Критерії оцінювання виконання лабораторної роботи

Якісні характеристики	Конкретизація вимог/бальна оцінка (з розрахунку max=4 бали)			
	Найвищий бал (Σ>3,5)	Добрі результати (Σ>2,5)	Задовільні (Σ>1,5)	Незадовільно (Σ<1,5)
Самостійність виконання завдань лабораторної роботи	Повністю самостійне виконання завдань з елементами інновації, креативності	Самостійне виконання завдань, деякі завдання виконуються з підказкою викладача	При виконанні завдання необхідна не однократна допомога викладача	Завдання не виконано, або не є власною роботою
	2,5	2	1,5	0
Дотримання графіку	Робота, або її ключові пункти виконані з випередженням строків	Робота виконана за графіком	Є незначне порушення строків виконання роботи (до тижня)	Робота у заплановані терміни не отримана
	1,2	1	0,8	0

Критерії оцінювання обговорення результатів на заключному семінарі

Якісні характеристики	Конкретизація вимог/бальна оцінка (з розрахунку max=4 бали)			
	Найвищий бал (Σ>3,5)	Добрі результати (Σ>2,5)	Задовільні (Σ>1,5)	Незадовільно (Σ<1,5)
Повнота та системність	Викладення отриманих знань повне, систематизоване, розширене додатковими елементами; відповідає вимогам навчальної програми; є зв'язок з іншими модулями курсу; поодинокі несуттєві помилки виправляються самостійно	Викладення отриманих знань повне, систематизоване, відповідає вимогам навчальної програми; помилки виправляються після підказки викладача	Викладення знань неповне, проблеми при виявленні причинно-наслідкових зв'язків і формулюванні висновків	Безсистемні виділення випадкових ознак вивченого; невміння виконувати найпростіші операції аналізу і синтезу; робити узагальнення, висновки
	2	1,5	1	0
Наочність результатів	Результати викладено логічно; захист (доклад, презентація) роботи проведено з застосуванням демонстраційних матеріалів (графіки, медіа-, кіно й т.п.), привернув увагу аудиторії. Представлено повний, детальний вербальний звіт з елементами креативності	В процесі викладання результатів роботи необхідна допомога викладача, захист роботи проведено згідно вимогам, демо-матеріали в мінімально необхідному обсязі. Вербальний звіт виконано згідно вимог	Захист результатів (доклад, виступ, презентація) роботи не викликало інтересу аудиторії, демо- матеріали відсутні. Вербальний звіт включає обмежену кількість інформації	Робота представлена без виступу. Звіт не надано
	1,5	1	0,8	0
Використання групових методів розробки	Застосовано методи групової роботи, виявлено якості лідера, ініціативність, креативність, навички комунікативного та перцептивного спілкування	В груповій роботі виявлено комунікаційні навички, вміння працювати в групі	В груповій роботі активність не висока	Участі у груповій роботі не приймав
	1,5	1	0,7	0

Критерії оцінювання фінальної роботи

Якісні характеристики	Конкретизація вимог/бальна оцінка (з розрахунку max=4 бали)			
	Найвищий бал (Σ>3,5)	**Добрі результати (Σ>2,5)**	**Задовільні (Σ>1,5)**	**Незадовільно (Σ<1,5)**
Повнота та системність виконання завдання	Викладення отриманих знань повне, систематизоване, розширене додатковими елементами; відповідає вимогам навчальної програми; є зв'язок з іншими модулями курсу; поодинокі несуттєві помилки виправляються самостійно	Викладення отриманих знань повне, систематизоване, відповідає вимогам навчальної програми; помилки виправляються після підказки викладача	Викладення знань неповне, проблеми при виявленні причинно-наслідкових зв'язків і формулюванні висновків	Безсистемні виділення випадкових ознак вивченого; невміння виконувати найпростіші операції аналізу і синтезу; робити узагальнення, висновки
	1,5	1	0,8	0
Оформлення фінальної роботи	Результати викладено логічно; захист (доклад, презентація) роботи проведено з застосуванням демонстраційних матеріалів (графіки, медіа-, кіно й т.п.), привернув увагу аудиторії. Представлено повний, детальний звіт з елементами креативності	В процесі викладання результатів роботи необхідна допомога викладача, захист роботи проведено згідно вимогам, демо-матеріали в мінімально необхідному обсязі. Звіт виконано згідно вимог	Захист результатів (доклад, виступ, презентація) роботи не викликало інтересу аудиторії, демо- матеріали відсутні. Звіт включає обмежену кількість інформації	Робота представлена без виступу. Звіт не надано
	1,5	1	0,8	0
Дотримання графіку	Фінальну роботу представлено з випередженням строків	Фінальну роботу представлено за графіком	Є незначне порушення строків представлення фінальної роботи	Фінальну роботу у заплановані терміни не отримано
	1,5	1,2	0,8	0

Навчальне електронне видання

Шерстюк В.Г.

ЕЛЕКТРОННИЙ НАВЧАЛЬНИЙ ПОСІБНИК

«ОСНОВИ РОЗРОБКИ КОМП'ЮТЕРНИХ ІГОР»

Для підготовки студентів
на першому (бакалаврському) рівні вищої освіти,
галузі знань 12 «Інформаційні технології»,
спеціальності 121 «Інженерія програмного забезпечення»,
освітньо-професійної програми «Програмна інженерія»

ISBN 978-617-7573-92-9 (електронне видання)

Підписано до видання 04.04.2019 р. Формат 60×84/8.
Гарнітура Times.
Ум. друк. арк. 22,17. Обл.-вид. арк. 23,84.
Замовлення № 1116.

Книжкове видавництво ФОП Вишемирський В. С.
Свідоцтво про внесення до Державного реєстру суб'єктів
видавничої справи: серія ХС № 48 від 14.04.2005 р.
видано Управлінням у справах преси та інформації
73000, Україна, м. Херсон, вул. Соборна, 2,
тел. (050) 133–10–13, e-mail: printvvs@gmail.com, vish_sveta@rambler.ru