МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ МОРСЬКИЙ УНІВЕРСИТЕТ

Кафедра «Технічної кібернетики й інформаційних технологій ім. проф. Р.В. Меркта »

Затверджено
НМК ННІ інформаційних
технологій та інноваційного підприємництва
Протокол № 4 від 19.02.2026 р.

Андрій ІВАНОВ

19 лютого 2026 р.

МЕТОДИЧНІ ВКАЗІВКИ

до розрахунково-графічного завдання з курсу

ОСНОВИ РОЗРОБКИ ІГОР

Для студентів спеціальності F3 «Комп'ютерні науки»
рівня підготовки «бакалавр»
денної та заочної форми навчання

Одеса – 2026

Методичні вказівки підготував кандидат географічних наук, доцент Даус Юрій Володимирович – викладач кафедри «Технічна кібернетика й інформаційні технології ім. проф. Р.В.Меркта» Одеського національного морського університету за діючою робочою програмою навчальної дисципліни «Основи розробки ігор».

Конспект лекцій схвалено кафедрою «Технічна кібернетика й інформаційні технології ім. проф. Р.В.Меркта» ОНМУ « 06 » лютого 2026 р., протокол № 14.

Розрахунково-графічне завдання з курсу «Основи розробки ігор»

Склав: доц каф. ТКйІТ,

к. геогр. н. Даус Ю.В.

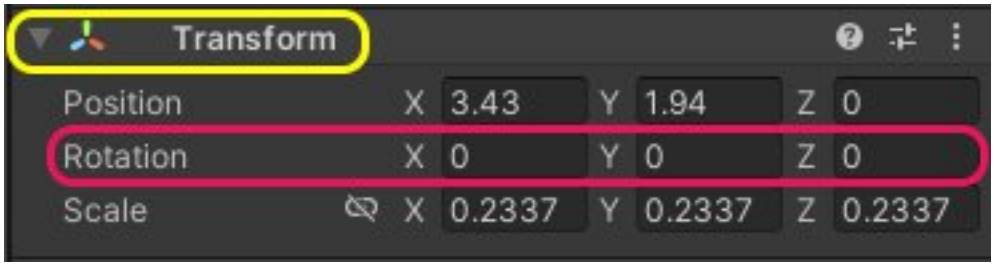
Метою розрахунково-графічного завдання (РГЗ) являється закріплення навичок програмування ігор у студентів за допомогою застосування набутих знань, відомих бібліотек, середовища UNITY та мови програмування C#.

Теоретична частина

Поворот і обертання об'єктів в Unity

Кожен розробник ігор знає, що без обертання і поворотів ігрових об'єктів неможливо зробити нормальну гру. І перед тим, як почати розбір повороту/обертання об'єкта, створимо ігровий об'єкт і назвемо його **Player**. Його ми і будемо обертати.

Як обертання об'єкта, так і обертання здійснюється шляхом зміни значень компонента Transform і його властивості Transform.rotation.



І почнемо цю статтю з обертання, оскільки вона простіша і зрозуміліша.

Обертання ігрового об'єкта

Для початку розберемося, що означає обертання? Обертання — це плавна зміна значень **Transform.rotation** навколо певних осей. Ось їх нам і потрібно змінити.

Для зміни цих значень нам допоможе метод Transform.Rotate. Він працює наступним чином:

```
void Update()
{
   transform.Rotate(0, 0, 1);
}
```

Цей скрипт буде обертати вашого персонажа по осі **Z** зі швидкістю 1. Ви також можете вибрати іншу вісь обертання, наприклад **X** або **Y**. Або ж збільшити швидкість обертання на 2.

```
void Update()
{
   transform.Rotate(2, 0, 0);
}
```

На цьому все! Як бачите, обертати ігрові об'єкти дуже просто. Тепер перейдемо до повороту, який за своєю суттю трохи важче.

Поворот ігрового об'єкта

Поворот персонажа передбачає його поворот на 180 градусів по одній з осей. Наприклад, дуже часто таке може знадобитися в 2д іграх, переважно платформерного типу. Тому в якості прикладу ми будемо використовувати 2д персонажа, якого потрібно розгорнути по горизонталі, тобто по осі X.

Попередній код обертання нам не допоможе, оскільки там ми обертали за допомогою спеціального методу, який додає до старих значень — нові. Тут же нам потрібно встановити конкретні значення.

Для того, щоб це зробити, необхідно прописати наступний код:

```
Vector3 rotate = transform.eulerAngles;
rotate.y = 180;
```

transform.rotation = Quaternion.Euler(rotate);

Давайте розберемо код:

- У першому рядку за допомогою методу Transform.eulerAngles ми отримуємо значення **roration** у вигляді кутів Ейлера.
- У другому рядку встановили значення **y** в 180, тим самим розгорнули об'єкт на 180 градусів.
- У третьому рядку елерівські значення перетворюємо в кватерніони і зберігаємо отримані дані.

Таким же чином ми можемо повернути об'єкт у нормальне положення:

Vector3 rotate = transform.eulerAngles;

rotate.y = 0;

transform.rotation = Quaternion.Euler(rotate);

Аналогічним чином ми можемо розгорнути наш об'єкт по будь-якій осі, змінивши **rotate.y** на **rotate.x** або на **rotate.z**.

Обертання та орієнтація в UNITY (офіційна документація)

Обертання в 3D-додатках зазвичай представлені одним із двох способів, кватерніонами або кутами Ейлера. Кожен має своє використання та недоліки. Unity використовує Quaternions внутрішньо, але показує значення еквівалентних кутів Ейлера в інспекторі, щоб полегшити вам редагування.

Різниця між кутами Ейлера та кватерніонами

Кути Ейлера

Кути Ейлера мають більш просте представлення, тобто три значення кута для X, Y та Z, які застосовуються послідовно. Щоб застосувати обертання Ейлера до певного об'єкта, кожне значення обертання застосовується по черзі, як обертання навколо відповідної осі.

- **Перевага**: кути Ейлера мають інтуїтивно зрозумілий формат "читабельний людиною", що складається з трьох кутів.
- **Перевага**: кути Ейлера можуть представляти обертання від однієї орієнтації до іншої через поворот більш ніж на 180 градусів

- **Обмеження**: кути Ейлера страждають від Gimbal Lock. При застосуванні трьох обертань по черзі, можливо, що перше або друге обертання призведе до того, що третя вісь буде спрямована в тому ж напрямку, що і одна з попередніх осей. Це означає, що "ступень свободи" був втрачений, тому що третє значення обертання не може бути застосоване навколо унікальної осі.

Кватерніони

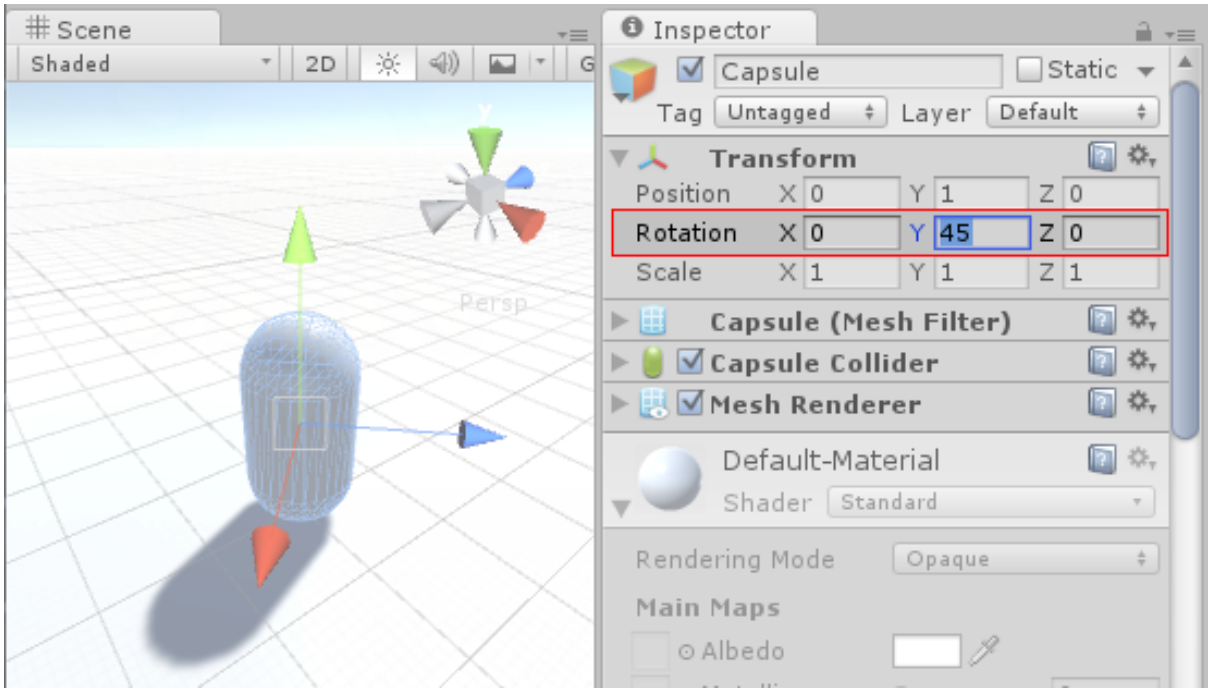
Кватерніони можна використовувати для представлення орієнтації або обертання об'єкта. Це представлення внутрішньо складається з чотирьох чисел (на які в Unity посилаються як x, y, z та w), однак ці числа не представляють кути або осі, і вам зазвичай ніколи не потрібно отримувати до них прямий доступ. Якщо ви не особливо зацікавлені в тому, щоб заглибитися в математику кватерніонів, вам дійсно потрібно лише знати, що кватерніон являє собою обертання в 3D-просторі, і вам ніколи не потрібно буде знати або змінювати властивості x, y і z.

Так само, як Вектор може представляти або позицію, або напрямок (де напрямок вимірюється від початку координат), кватерніон може представляти або орієнтацію, або обертання - де обертання вимірюється від обертального "походження" або "Ідентичність". Саме тому, що обертання вимірюється таким чином - від однієї орієнтації до іншої - кватерніон не може представляти обертання понад 180 градусів.

- **Перевага**: Ротації Quaternion не страждають від Gimbal Lock.
- **Обмеження**: Один кватерніон не може представляти обертання, що перевищує 180 градусів у будь-якому напрямку.
- **Обмеження**: Чисельне представлення кватерніону не є інтуїтивно зрозумілим.

У Unity всі обертання Game Object зберігаються внутрішньо як Quaternions, оскільки переваги переважують обмеження.

Однак в інспекторі перетворення ми відображаємо обертання за допомогою кутів Ейлера, тому що це легше зрозуміти та відредагувати. Нові значення, введені в інспектор для обертання ігрового об'єкта, перетворюються "під капотом" на нове значення обертання Quaternion для об'єкта.



Обертання ігрового об'єкта відображається та редагується як кути Ейлера в інспекторі, але зберігається внутрішньо як Quaternion

Як побічний ефект, в інспекторі можна ввести значення, скажімо, **X: 0, Y: 365, Z: 0** для обертання ігрового об'єкта. Це значення, яке неможливо представити як кватерніон, тому, коли ви натиснете Play, ви побачите, що значення обертання об'єкта змінюються на **X: 0, Y: 5, Z: 0** (або близько того). Це пов'язано з тим, що обертання було перетворено на Quaternion, який не має концепції "Повне обертання на 360 градусів плюс 5 градусів", а замість цього просто було налаштовано так само, як і результат обертання.

Наслідки для написання сценаріїв

Маючи справу з обробкою ротацій у ваших скриптах, ви повинні використовувати клас Quaternion та його функції для створення та зміни значень обертання. Є деякі ситуації, коли допустимо використовувати кути Ейлера, але ви повинні мати на увазі: - Ви повинні використовувати функції класу Quaternion, які мають справу з кутами Ейлера - Отримання, зміна та повторне застосування значень Ейлера з обертання може спричинити ненавмисні побічні ефекти.

Створення та маніпулювання кватерніонами безпосередньо

Клас Quaternion Unity має ряд функцій, які дозволяють створювати та маніпулювати обертаннями без необхідності використовувати кути Ейлера взагалі. Наприклад:

Створення:

- Quaternion.LookRotation
- Кватерніон.Вісь Кута
- Quaternion.FromToRotation

Маніпуляція:

- Кватерніон.Слерп
- Quaternion.Зворотний
- Quaternion.RotateTowards
- Трансформувати.Оворот і Трансформувати.Оворот Навколо

Однак іноді бажано використовувати кути Ейлера у ваших скриптах. У цьому випадку важливо зазначити, що ви повинні зберігати свої кути в змінних і використовувати їх лише для того, щоб *застосувати* їх як кути Ейлера до обертання. Хоча можна отримати кути Ейлера *з* кватерніону, якщо ви отримаєте, зміните та повторно застосуєте, виникнуть проблеми.

Ось кілька прикладів **помилок**, **які** зазвичай допускаються, використовуючи гіпотетичний приклад спроби повернути об'єкт навколо осі X зі швидкою 10 градусів за секунду. Це те, чого слід ***уникати***:

```
// rotation scripting mistake #1
// the mistake here is that we are modifying the x value of a quaternion
// this value does not represent an angle, and will not produce desired results

void Update () {

    var rot = transform.rotation;
    rot.x += Time.deltaTime * 10;
    transform.rotation = rot;

}
// rotation scripting mistake #2
```

```
    // the mistake here is that we are reading, modifying then writing the Euler
    // values from a quaternion. Because these values calculated from a
Quaternion,
    // each new rotation may return very different Euler angles, which may suffer
from gimbal lock.

    void Update () {

        var angles = transform.rotation.eulerAngles;
        angles.x += Time.deltaTime * 10;
        transform.rotation = Quaternion.Euler(angles);

    }
```

І ось приклад ***правильного*** використання кутів Ейлера в скрипті:

```
    // rotation scripting with Euler angles correctly.
    // here we store our Euler angle in a class variable, and only use it to
    // apply it as a Euler angle, but we never rely on reading the Euler back.

    float x;
    void Update () {

        x += Time.deltaTime * 10;
        transform.rotation = Quaternion.Euler(x,0,0);

    }
```

Наслідки для анімації

Багато пакетів 3D-авторингу, а також власне вікно анімації Unity, дозволяють використовувати кути Ейлера для визначення обертань під час анімації.

Ці значення обертань часто можуть перевищувати діапазон, виражений кватерніонами. Наприклад, якщо об'єкт повинен обертатися на 720 градусів на місці, це може бути представлено кутами Ейлера X: 0, Y: 720, Z:0. Але це просто не можна представити значенням кватерніону.

Вікно анімації Unity

У власному вікні анімації Unity є параметри, які дозволяють вказати, як обертання має бути інтерпольовано - за допомогою інтерполяції Quaternion або Euler. Вказуючи інтерполяцію Ейлера, ви говорите Unity, що вам потрібен повний діапазон руху, визначений кутами. Однак з обертанням Quaternion ви кажете, що просто хочете, щоб обертання закінчилося на певній орієнтації, і Unity використовуватиме інтерполяцію Quaternion і обертатиме на найкоротшу відстань, щоб дістатися туди. Див. Використання кривих анімації для отримання додаткової інформації про це.

Зовнішні джерела анімації

При імпорті анімації із зовнішніх джерел ці файли зазвичай містять ротаційну анімацію ключових кадрів у форматі Ейлера. Поведінка Unity за замовчуванням полягає в тому, щоб перевибірка цих анімацій і генерувати новий ключовий кадр Quaternion для кожного кадру в анімації, намагаючись уникнути будь-яких ситуацій, коли обертання між ключовими кадрами може перевищувати дійсний діапазон Quaternion.

Наприклад, уявіть собі два ключові кадри, 6 кадрів один від одного, зі значеннями X як 0 на першому ключовому кадрі і 270 на другому ключовому кадрі. Без повторної вибірки інтерполяція кватерніону між цими двома ключовими кадрами буде обертатися на 90 градусів у протилежному напрямку, тому що це найкоротший спосіб дістатися від першої орієнтації до другої орієнтації. Однак, перевибірка та додавання ключового кадру на кожному кадрі, тепер між ключовими кадрами є лише 45 градусів, тому обертання буде працювати правильно.

Все ще є деякі ситуації, коли - навіть при повторній вибірці - кватерніонне представлення імпортованої анімації може не відповідати оригіналу. З цієї причини в Unity 5.3 і далі є можливість вимкнути повторну вибірку анімації, щоб замість цього ви могли використовувати оригінальні ключові кадри анімації Ейлера під час виконання. Для отримання додаткової інформації див. Імпорт анімації обертань кривих Ейлера.

Аудіо.

Гра була б неповною без будь-якого звуку, будь то музичний фон або звукові ефекти. Аудіосистема Unity гнучка і потужна. Вона може імпортувати більшість стандартних аудіоформатів і має складні функції для відтворення звуків у 3D-просторі, з додатковими ефектами, такими як використання відлуння та фільтрації. Unity також може записувати аудіо з будь-якого доступного мікрофона на комп'ютері користувача, для використання під час гри або для зберігання та передачі.

Основна теорія

У реальному житті звуки видаються об'єктами і чують слухачі. Сприйняття звуку залежить від ряду факторів. Слухач може визначити, звідки приблизно йде звук, а за гучністю і якістю звуку визначити приблизну відстань до джерела. Джерело звуку, що швидко рухається (як бомба, що падає, або поліцейська машина, що проїжджає) буде змінюватися по висоті під час руху, в результаті ефекту Допплера. Крім того, середовище впливатиме на відбиття звуку. Так що голос у печері матиме відлуння, а на відкритому повітрі — ні.

Аудіоджерела (Source) і слухач (Listener)

Щоб імітувати ефекти розташування, Unity вимагає, щоб звуки виходили з компонентів Audio Source, прикріплених до об'єктів. Потім, випромінювані звуки «ловляться» компонентом Audio Listener, прикріпленим до іншого об'єкта, найчастіше, до камери. Потім Unity може імітувати ефекти відстані і просторового положення джерела від слухача і відтворювати їх для користувача відповідним чином. Відносна швидкість об'єктів джерела і слухача також може бути використана для імітації ефекту Допплера для додаткової реалістичності.

Unity не може обчислити відлуння тільки виходячи з геометрії сцени, але ви можете імітувати його, додавши аудіофільтри (Audio Filters) до об'єктів. Наприклад, ви могли б застосувати фільтр Echo до звуку, який призначений для звучання з печери. У випадках, коли об'єкти можуть рухатися всередину і назовні з області сильного відлуння, ви можете додати в сцену зону реверберації (Reverb Zone). Наприклад, у вашій грі автомобілі можуть проїжджати по тунелю. Якщо ви розмістите зону реверберації всередині тунелю, звуки двигунів автомобілів почнуть лунати в момент в'їзду в тунель і відлуння припиниться, коли вони будуть виїжджати з іншого боку тунелю,

Unity **Audio Mixer** дозволяє змішувати різні джерела звуку, застосовувати до них ефекти та виконувати мастеринг.

На сторінках керівництва з Audio Source, Audio Listener, аудіоефектів і зон реверберації можна знайти додаткову інформацію про багато налаштувань і опцій, доступних для отримання потрібних ефектів.

Робота з аудіоактивами

Unity може імпортувати файли у форматах **AIFF**, **WAV**, **MP3** та **Ogg** тим же способом, що й інші ресурси, просто перетягуючи файли на панель Project. Імпорт аудіофайлу створює аудіокліп (Audio Clip), який можна перетягнути на джерело звуку (Audio Source) або використовувати зі скрипта. Сторінка довідки Audio Clip містить більше інформації про параметри імпорту, доступні для аудіофайлів.

Для музики Unity також підтримує трекерні модулі, які використовують короткі аудіо-семпли як «інструменти», які надалі впорядковуються в мелодію. Трекерні модулі можуть бути імпортовані з **.xm**, **.mod**, **.it**, і **.s3m** файлів, і використовуватися як звичайні аудіокліпи.

Запис аудіо

Unity може отримати доступ до мікрофонів комп'ютера зі скрипта та створювати аудіокліпи із запису. Клас Microphone надає простий API для пошуку доступних мікрофонів, для запиту їх можливостей, а також для початку і закінчення запису. Сторінка довідки для класу Microphone містить більше інформації та прикладів коду для запису звуку.

Аудіофайли

Аналогічно мешам або текстурам, робочий процес для аудіофайлів (аудіофайл) розроблений для плавної та безпроблемної роботи. Unity може

імпортувати файли майже будь-якого формату, але є кілька корисних деталей, які бажано знати при роботі з аудіофайлами.

Оскільки аудіодані Unity 5.0 відокремлені від фактичних AudioClips. AudioClips просто посилаються на файли, що містять аудіодані, і в імпортері AudioClip є різні комбінації опцій, які визначають, як завантажуються кліпи під час виконання. Це означає, що у вас є велика гнучкість у вирішенні того, які аудіоактиви слід постійно зберігати в пам'яті (тому що ви, можливо, не зможете передбачити, як часто або як швидко вони будуть грати, тобто кроки, зброя та удари), тоді як інші активи можуть завантажуватися на вимогу або поступово, коли гравець просувається через рівень (мова, фонова музика, петлі атмосфери тощо).

Коли аудіо кодується в Unity, основними параметрами його зберігання на диску є або *PCM*, *ADPCM* або *Compressed*. Крім того, існує кілька форматів, специфічних для платформи, але вони працюють подібним чином. Unity підтримує найпоширеніші формати для імпорту аудіо (див. список нижче) і імпортує аудіофайл при додаванні його до проєкту. Режим за замовчуванням - *Стиснутий*, де аудіодані стискаються або за допомогою Vorbis/MP3 для автономних та мобільних платформ, або HEVAG/XMA для PS Vita / XBox One.

Дивіться документацію AudioClip для детального опису форматів стиснення та інших опцій, доступних для кодування аудіоданих.

Будь-який аудіофайл, імпортований в Unity, доступний зі скриптів як екземпляр **Audio Clip**, що забезпечує спосіб для ігрового часу виконання аудіосистеми для доступу до закодованих аудіоданих. Гра може отримати доступ до метаінформації про аудіодані через AudioClip ще до того, як фактичні аудіодані були завантажені. Це можливо, оскільки процес імпорту витягнув різні біти інформації, такі як довжина, кількість каналів та частота дискретизації з закодованих аудіоданих, і зберіг їх у AudioClip. Це може бути корисно, наприклад, при створенні автоматичних діалогів або систем послідовності музики, оскільки музичний движок може використовувати інформацію про довжину для планування відтворення музики перед фактичним завантаженням даних. Це також допомагає зменшити використання пам'яті, зберігаючи в пам'яті лише ті аудіокліпи, які потрібні одночасно.

Підтримувані формати

Формат	Розширення
MPEG рівень 3	.mp3
Огг Ворбіс	.ogg
Хвиля Microsoft	.wav
Формат файлу обміну аудіо	.aiff / .aif
Модуль Ultimate Soundtracker	.мод
Модуль імпульсного трекера	.it
Модуль Scream Tracker	.s3m
Модуль FastTracker 2	.xm

Дивіться огляд аудіо для детальної інформації про використання звуків в Unity.

Трекерні модулі

Трекерні модулі по суті просто набір аудіофрагментів, моделювання, аранжування та зведення яких проведено на програмному рівні. Концепція була представлена в 80-х роках (в основному в поєднанні з комп'ютером Amiga) і ця система була дуже популярна з ранніх етапів ігрової індустрії і демо-культури.

Файли трекерних модулів багато в чому схожі на MIDI файли. Є багато треків, що містять інформацію про те, коли інструменти грають і з якими показниками гучності і висоти. І таким чином мелодія і ритм вихідного треку можуть бути відтворені. Тим не менш, у MIDI є мінус, який полягає в тому, що звуки залежать від набору звуків, доступних у звуковому обладнанні, таким чином MIDI музика може звучати по-різному на різних комп'ютерах. На відміну від цього, самі трекерні модулі включають в себе високоякісні фрагменти PCM, в результаті чого виходить подібний результат незалежно від використовуваного звукового обладнання.

Підтримувані формати

Unity підтримує 4 найпоширеніші формати файлів модулів, а саме Impulse Tracker (**.it**), Scream Tracker (**.s3m**), формат файлу Extended Module (**.xm**) і оригінальний формат файлу Module (**.mod**).

Користь від використання трекерних модулів

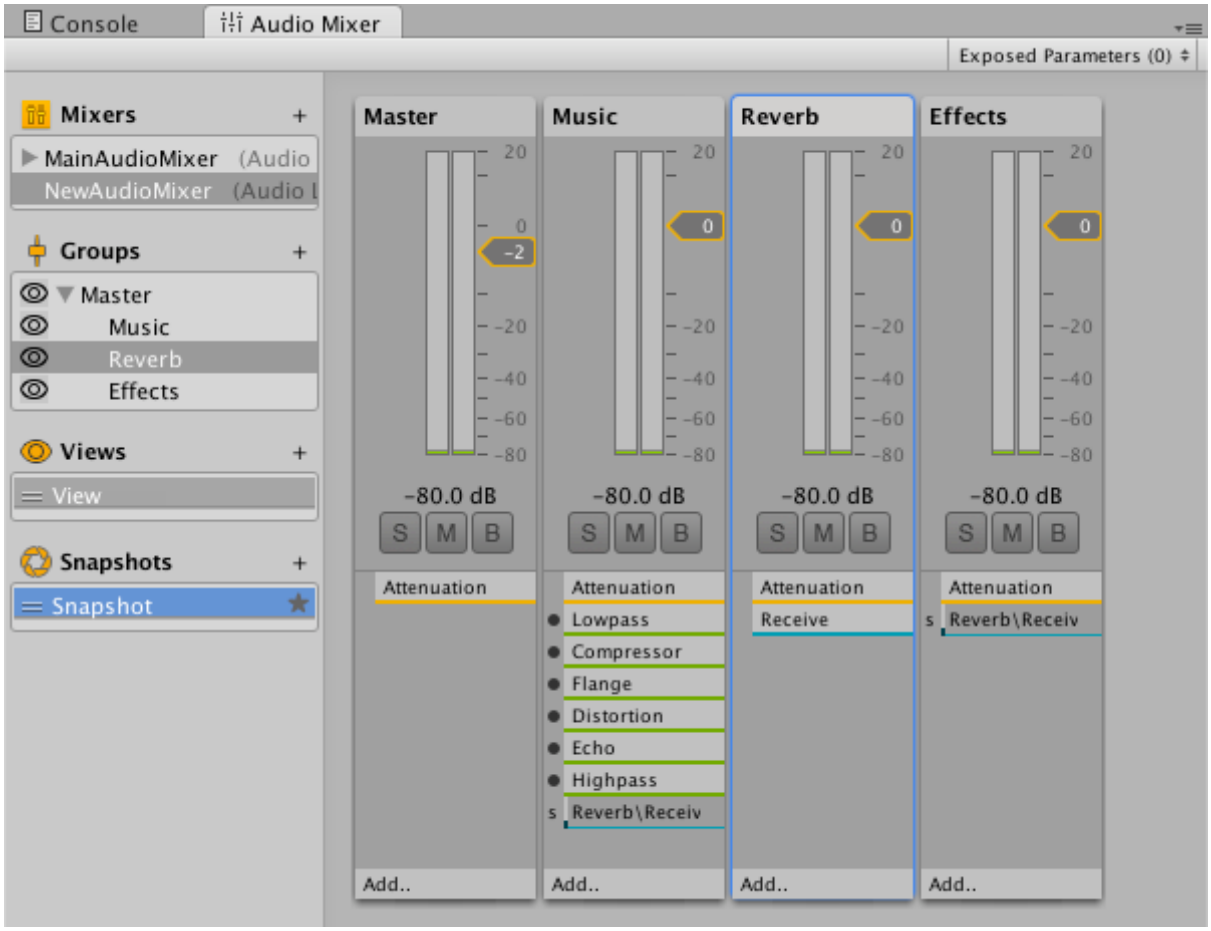
Трекерні модулі відрізняються від популярних форматів PCM (**.aif**, **.wav**, **.mp3** і **.ogg**) тим, що можуть мати дуже маленький розмір без втрати якості звуку. Один звуковий фрагмент може змінюватися за гучністю і висотою (також можуть застосовуватися й інші ефекти), таким чином, він виступає в ролі «інструменту», яким можна зіграти мелодію без необхідності записувати повну мелодію окремим фрагментом. В результаті трекерні модулі добре підходять для ігор, де потрібен музичний супровід, але завантаження великих файлів може бути проблемою.

Аудіо Мікшер

Unity Audio Mixer дозволяє змішувати різні джерела звуку, застосовувати до них ефекти та виконувати мастеринг.

Вікно аудіомікшера

У вікні відображається аудіомікшер, який в основному є деревом груп аудіомікшерів. Група аудіомікшерів - це, по суті, суміш аудіо, сигнальний ланцюг, який дозволяє застосовувати загасання гучності та корекцію висоти тону; це дозволяє вставляти ефекти, які обробляють аудіосигнал, і змінювати параметри ефектів. Також існує механізм надсилання та повернення для передачі результатів з однієї шини в іншу.



Аудіомікшер

Аудіомікшер є активом. Ви можете створити один або кілька аудіомікшерів і мати більше одного активного в будь-який час. Аудіомікшер завжди містить головну групу. Потім можна додати інші групи, щоб визначити структуру змішувача.

Як це працює

Ви направляєте вихід [джерела звуку](#) до групи в межах аудіомікшера. Потім ефекти будуть застосовані до цього сигналу.

Вихід аудіомікшера може бути направлений в будь-яку іншу групу в будь-якому іншому аудіомікшері в сцені, що дозволяє вам з'єднати ряд аудіомікшерів у сцені для створення складної маршрутизації, обробки ефектів та застосування знімків.

Знімки

Ви можете захопити налаштування всіх параметрів у групі як знімок. Якщо ви створюєте список знімків, ви можете переходити між ними в ігровому процесі, щоб створити різні настрої або теми.

Ухилення

Ducking дозволяє змінити ефект однієї групи на основі того, що відбувається в іншій групі. Прикладом може бути зменшення фонового шуму навколишнього середовища, поки відбувається щось інше.

Погляди

Можна налаштувати різні перегляди. Ви можете вимкнути видимість певних груп у мікшері та встановити це як перегляд. Потім ви можете переходити між переглядами за потребою.

Елементи UI

Rich Text

Текст для елементів інтерфейсу користувача та текстових сіток може містити різні стилі та розміри шрифту. Класи Text, GUIStyle і TextMesh мають параметр **Rich Text** , який наказує Unity шукати теги розмітки в тексті. Функція Debug.Log також може використовувати ці теги розмітки для покращення звітів про помилки з коду. Теги не відображаються, але вказують на зміни стилю, які слід застосувати до тексту.

Формат розмітки

Система розмітки тексту в Unity була створена на основі HTML, однак сувора сумісність зі стандартним HTML не передбачається. Основна ідея полягає в тому, що фрагменти тексту можна укладати в пару тегів, що узгоджуються один з одним:-

We are not amused.

Як показує приклад, теги — це лише фрагменти тексту всередині символів «кутових дужок» <i >.

Ви розміщуєте *початковий* тег на початку розділу. Текст всередині тегу вказує на його ім'я (яке в даному випадку є просто **b**).

Ви розміщуєте ще один тег у кінці розділу. Це *закриваючий* тег. Він має таке ж ім’я, як і початковий тег, але перед ім’ям стоїть коса риска /. Кожен відкриваючий тег повинен мати відповідний закриваючий тег. Якщо ви не *закриєте* початковий тег, він буде відображено як звичайний текст.

Теги не відображаються користувачеві безпосередньо, але інтерпретуються як інструкції щодо стилізації тексту, який вони містять. Тег b, використаний у наведеному вище прикладі, застосовує жирний шрифт до слова «не», тому текст відображається на екрані як:

We are not amused

> We are **not** amused.

Розмічений фрагмент тексту (включаючи теги, що обертають його) також називається **елементом** .

Вкладені елементи

Існує можливість застосовувати більше одного стилю до фрагмента тексту за допомогою вкладення одних елементів в інші.

We are definitely not amused

Тег застосовує курсив, тому це буде представлено на екрані як

Нам ***точно не*** весело

> We are ***definitely not*** amused

Зверніть увагу на порядок закриваючих тегів, який є зворотним до порядку відкриття тегів. Причина цього, мабуть, більш зрозуміла, якщо врахувати, що внутрішні теги не обов'язково охоплюють весь текст зовнішнього елемента

We are absolutely definitely not amused

дає наступний результат:

Нам це **абсолютно *точно* не** весело

> We are **absolutely *definitely* not** amused

Параметри тегів

Деякі теги мають простий ефект «все або нічого» на текст, але інші можуть допускати варіації. Наприклад, тег **кольору** повинен знати, який колір застосувати. Подібна інформація додається до тегів за допомогою **параметрів** :-

We are <color=green>green</color> with envy

Що дає такий результат:

Зауважте, що кінцевий тег не містить значення параметра. За бажанням значення можна взяти в лапки, але це не обов'язково.

Параметри тегів не можуть містити пробіли. Наприклад:

We are <color = green>green</color> with envy

не працює через пробіли з обох боків від =символу.

Підтримувані теги

У наведеному нижче списку описано всі теги стилів, які підтримує Unity.

Тег	***Опис***	***Шістнадцятковий код***	***Примітки***
b	Виводить текст жирним шрифтом.	We are not amused.	
i	Виділяє текст курсивом.	We are usually not amused.	

Тег	*Опис*	*Шістнадцятковий код*	*Примітки*
розмір	Встановлює розмір тексту відповідно до значення параметра, заданого в пікселях.	We are <size=50>largely</size> unaffected.	Хоча цей тег доступний для Debug.Log, ви побачите, що інтервал між рядками на панелі вікна та в консолі виглядає дивно, якщо встановлено занадто великий розмір.
колір	Встановлює колір тексту відповідно до значення параметра. Колір можна вказати в традиційному форматі HTML. #rrggbbaa …де літери відповідають парам шістнадцяткових цифр, що позначають значення червоного, зеленого, синього та альфа (прозорість) кольору. Наприклад, блакитний із повною непрозорістю буде	We are <color=#ff0000ff>colorfully</color> amused	Інший варіант - використовувати назву кольору. Це легше зрозуміти, але, природно, діапазон кольорів обмежений і завжди передбачається повна непрозорість. <color=cyan>some text</color>Доступні назви кольорів наведено в таблиці нижче .

Тег	*Опис*	*Шістнадцятковий код*	*Примітки*
	визначено color=#00ffffff... Ви можете вказати шістнадцяткові значення у верхньому чи нижньому регістрі; #FF0000e квівалентно #ff0000.		
матеріал	Це корисно лише для текстових сіток і відображає частину тексту з матеріалом, визначеним параметром. Значення є індексом у масиві матеріалів текстової сітки, як показано інспектором.	We are <material=2>texturally </material> amused	
чотирикутник	Це корисно лише для текстових сіток і рендерить зображення в рядку з текстом. Він приймає параметри, які вказують матеріал	<quad material=1 size=20 x=0.1 y=0.1 width=0.5 height=0.5>	Це вибирає матеріал у позиції в масиві матеріалу рендерера та встановлює висоту зображення на 20 пікселів. Прямокутна область

Тег	*Опис*	*Шістнадцятковий код*	*Примітки*
	для використання для зображення, висоту зображення в пікселях і ще чотири, які позначають прямокутну область зображення для відображення. На відміну від інших тегів, quad не оточує фрагмент тексту, тому немає кінцевого тегу - символ косої риски розміщується в кінці початкового тегу, щоб вказати, що він «самозакриваєтьс я».		зображення починається з заданих значень x, y, ширини та висоти, які всі подані як частка немасштабованої ширини та висоти текстури.

Підтримувані кольори

У наведеній нижче таблиці наведено кольори, для яких можна використовувати назву замість шістнадцяткового тегу в тегу <color>форматованого тексту.

Назва кольору	***Шістнадцятковий код***	***приклад***
аква (те саме, що блакитний)	#00ffffff	
чорний	#000000ff	
блакитний	#0000ffff	
коричневий	#a52a2aff	
блакитний (те саме, що аква)	#00ffffff	
темно-синій	#0000a0ff	
фуксія (те саме, що пурпурний)	#ff00ffff	
зелений	#008000ff	
сірий	#808080ff	
блакитний	#add8e6ff	
вапно	#00ff00ff	
пурпурний (те саме, що фуксія)	#ff00ffff	

Назва кольору	***Шістнадцятковий код***	***приклад***
бордовий	#800000ff	
флот	#000080ff	
оливковий	#808000ff	
помаранчевий	#ffa500ff	
фіолетовий	#800080ff	
червоний	#ff0000ff	
срібло	#c0c0c0ff	
чирок	#008080ff	
білий	#ffffffff	
жовтий	#ffff00ff	

GUI редактор

Форматований текст вимкнено за замовчуванням у системі графічного інтерфейсу редактора, але його можна ввімкнути явно за допомогою спеціального GUIStyle . Для властивості richText має бути встановлено значення true, а стиль потрібно передати відповідній функції GUI:

```
GUIStyle style = new GUIStyle ();

style.richText = true;

GUILayout.Label("<size=30>Some <color=yellow>RICH</color>
text</size>",style);
```

Canvas (Полотно)

Canvas (полотно) – це область, всередині якої знаходяться всі елементи UI (інтерфейсу користувача). Полотно – це ігровий об'єкт (Game Object) з доданим до нього компонентом Canvas. Всі елементи UI повинні бути дочірніми цьому Canvas.

Коли ви створюєте новий елемент UI, такий як зображення, використовуючи меню **GameObject > UI > Image** , разом з ним автоматично створюється і Canvas, якщо до цього на сцені його ще не було. Елемент UI створюється дочірнім Canvas.

Область Canvas відображається у вигляді прямокутника у вікні Scene View. Це полегшує процес розташування елементів UI без потреби бачити ігрове вікно (Game View).

Canvas використовує EventSystem object to help Messaging System.

Порядок відображення елементів

Елементи UI на Canvas виникають у тому порядку, як вони розташовані в ієрархії. Перший дочірній елемент малюється першим, другий – за ним і таке інше. Якщо два елементи UI накладаються один на одного, доданий пізніше буде поверх того, що було додано раніше.

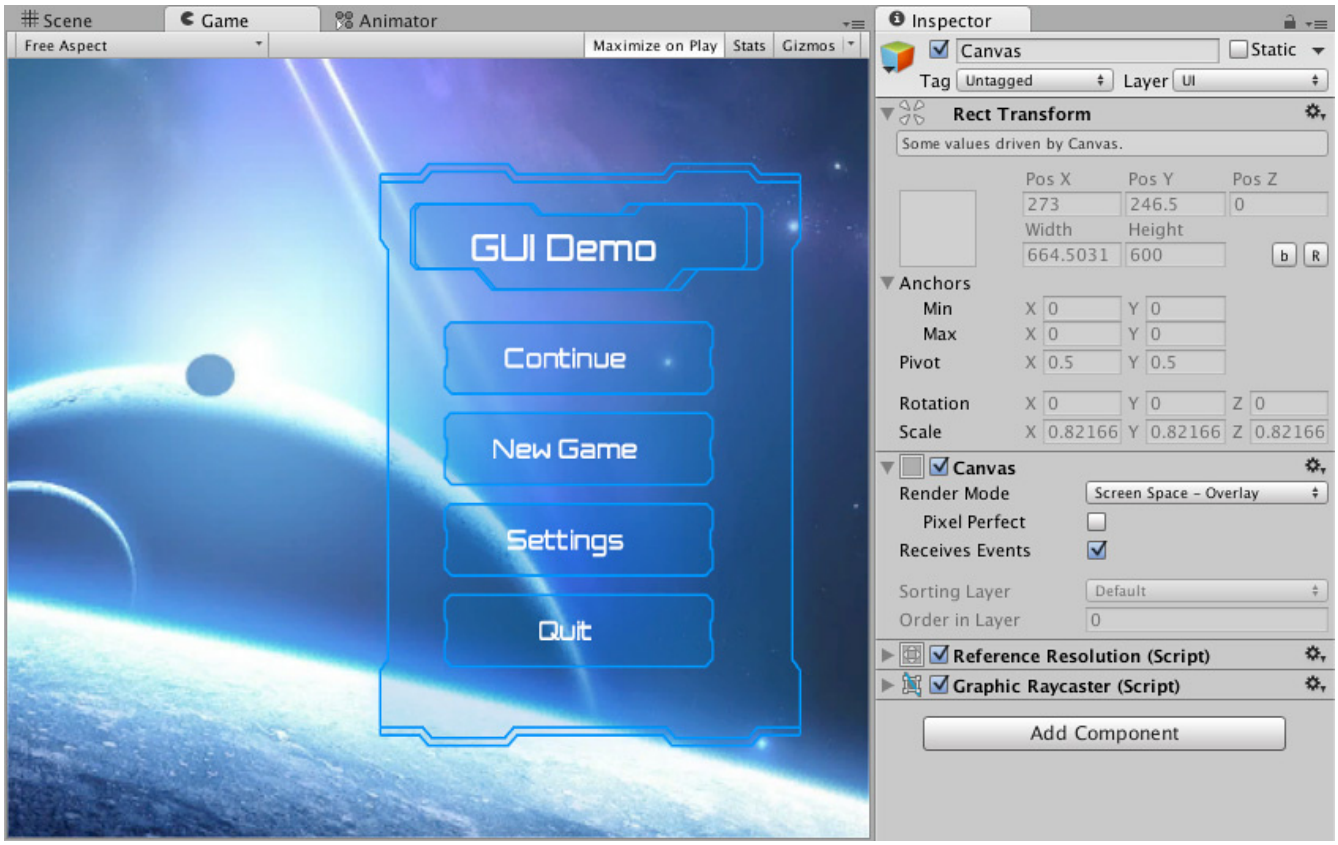
Щоб змінити те, який елемент буде поверх інших, просто поміняйте місцями елементи в ієрархії шляхом перетягування. Порядком можна керувати за допомогою скриптингу, використовуючи такі методи компонента Transform: SetAsFirstSibling, SetAsLastSibling і SetSiblingIndex.

Режими відображення

У полотна є параметр **Render Mode** , який визначає, де воно відображатиметься: у просторі екрану (screen space) чи ігрового світу (world space).

Простір екрану - Перекриття (Screen Space - Overlay)

Цей режим відображає елементи інтерфейсу на екрані поверх сцени. Якщо змінюється розмір екрана або його роздільна здатність, полотно автоматично прийме потрібний розмір разом із ним.

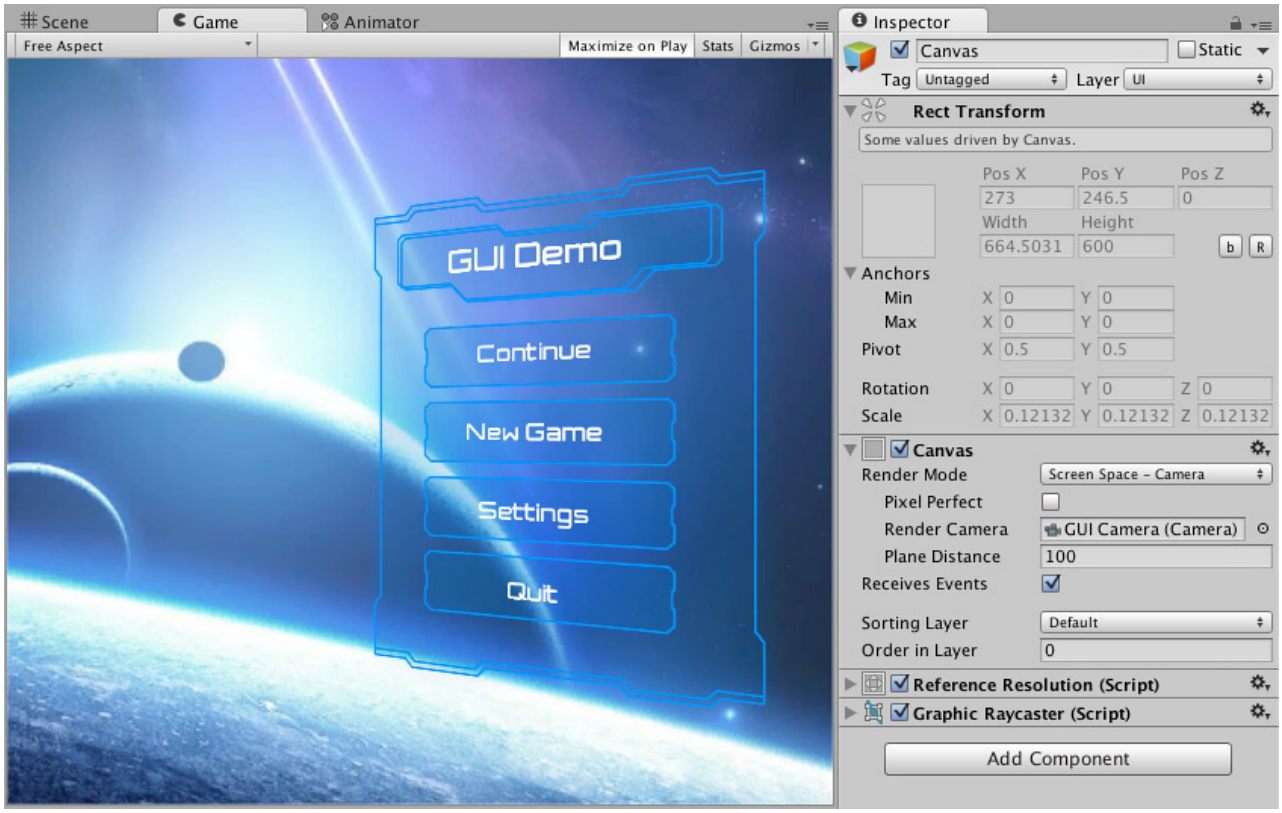


UI in screen space overlay canvas

Простір екрану - Камера (Screen Space - Camera)

Це є подібним до **Space - Overlay** , але в цьому режимі режиму Canvas є розташований на відстані довжини в конкретній **камері** . У UI елементи є зображені на цій камері, яка означає, що камера налаштовує на з'єднання з

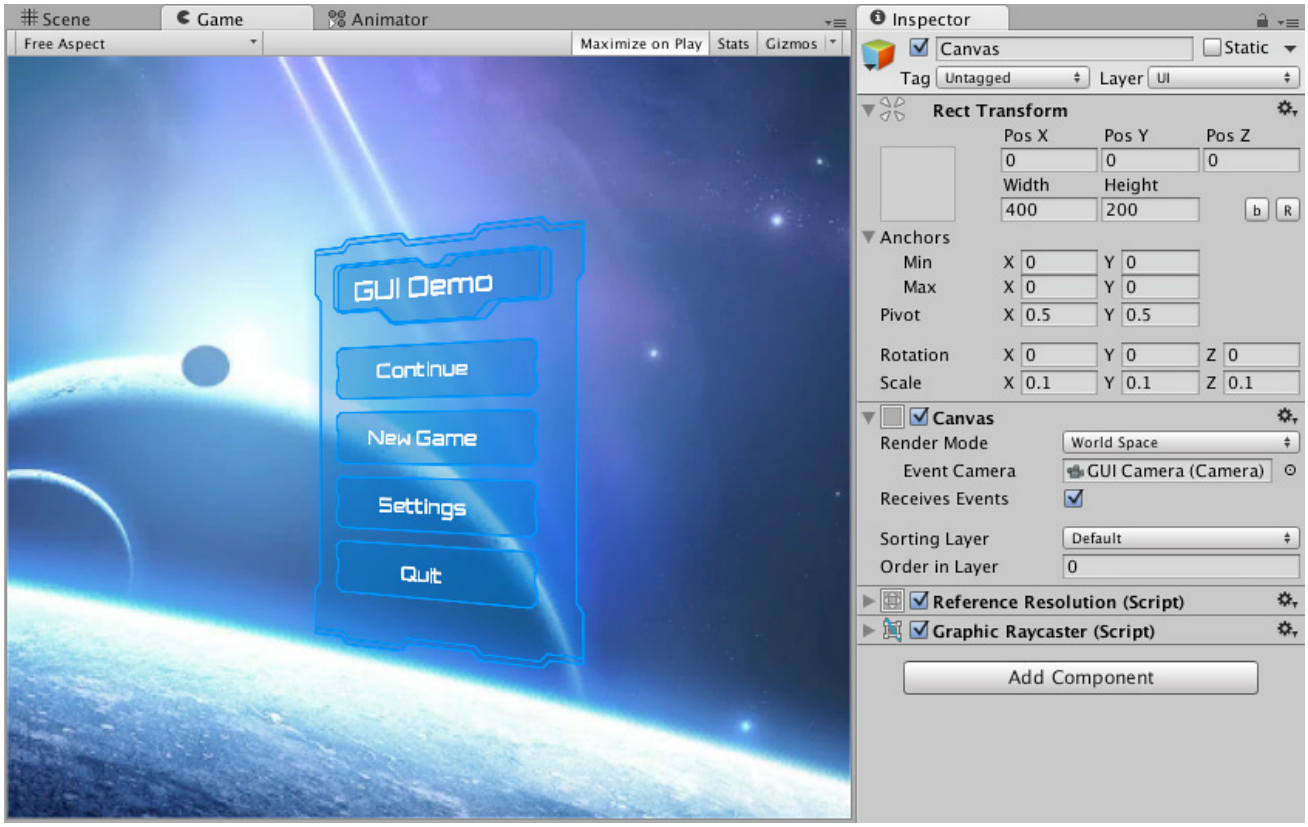
UI. Якщо Camera is set to **Perspective** , UI елементи будуть rendered with perspective, і сума perspective distortion може бути керована Camera **Field of View** . Якщо на екрані є відповідь, зміна рішення, або камера frustum зміни, Canvas буде автоматично змінити розмір, щоб зробити як добре.



UI in screen space camera canvas

Простір ігрового світу (World Space)

При цьому режим Canvas поводиться так само, як і будь-який інший об'єкт на сцені. Розмір Canvas може бути заданий вручну за допомогою Rect Transform, а елементи інтерфейсу відображатимуться перед або за іншими об'єктами на сцені, залежно від їхнього тривимірного розташування. Цей режим зручний тим інтерфейсів, які передбачаються як частина ігрового світу (diegetic interfaces).

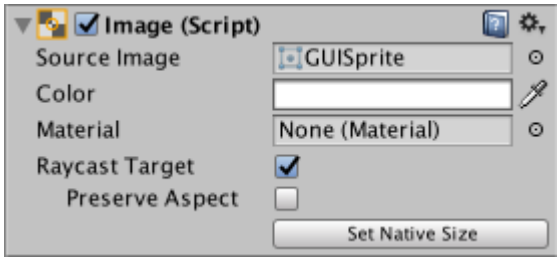


UI in world space canvas

Basic Layout

У цьому розділі ми розглянемо, як ви можете розташувати елементи інтерфейсу користувача відносно полотна та один одного. Якщо ви хочете перевірити себе під час читання, ви можете створити зображення за допомогою меню **GameObject -> UI -> Image** .

Зображення

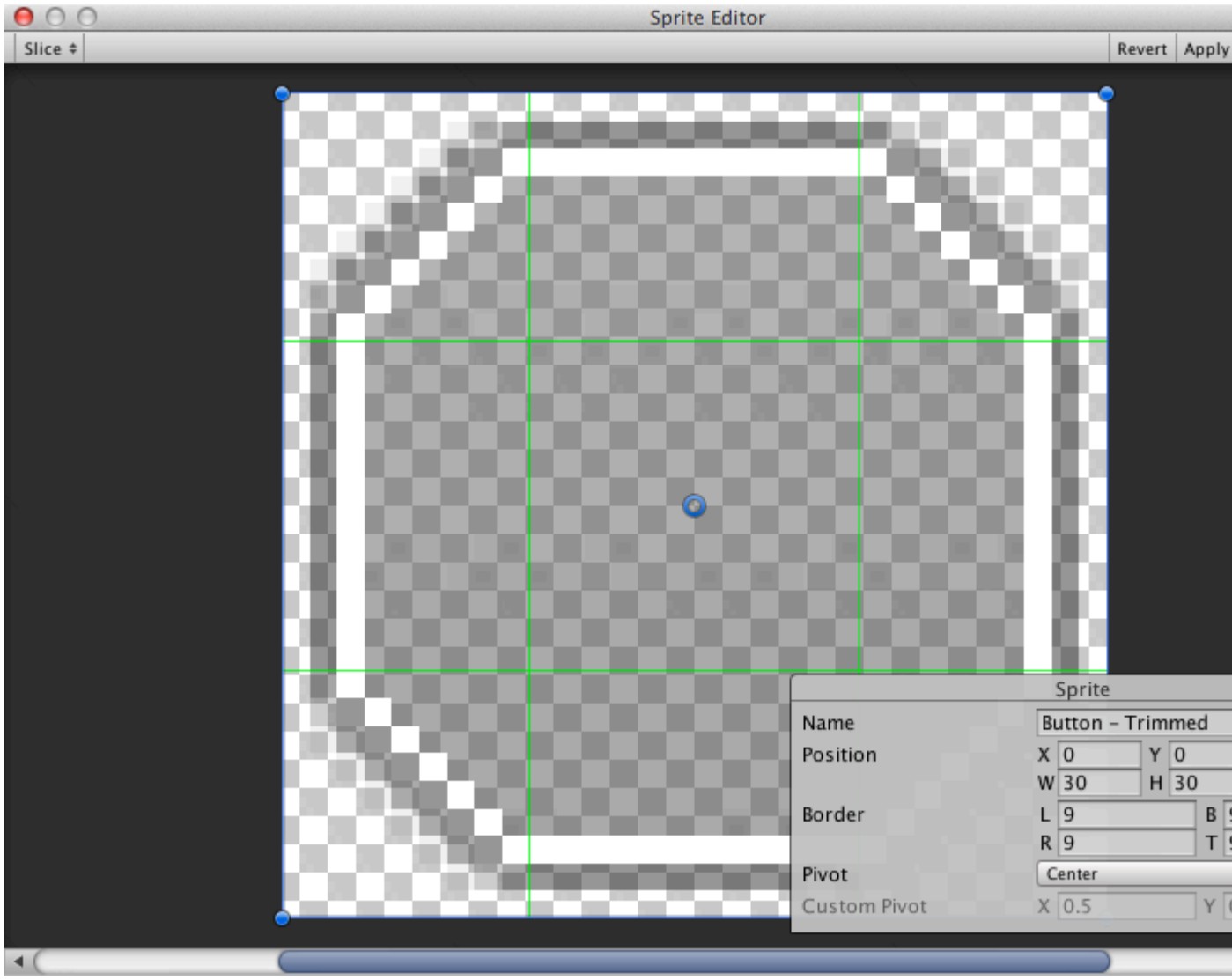


Перемикач (Toggle) має компоненти Rect Transform та **Image** (зображення). Спрайт може бути доданий до Image у полі Target Graphic, яке колір заданий у полі Color. Також до компоненту Image може бути доданий матеріал. Поле Image Type визначає, яким чином доданий спрайт відображатиметься, можливі варіанти:

- **Simple** – масштабує весь спрайт рівноцінно.
- **Sliced** – використовує поділ спрайту на 3x3 так, щоб масштабування не спотворювало кути, а розтягнута була лише центральна частина.
- **Tiled** - схоже на Sliced, але замість розтягування центральної частини вона повторюється (заповнюється тайлами). Для спрайтів без будь-яких кордонів весь спрайт буде заповнений тайлами.
- **Filled** - відображає спрайт так, як це робить Simple, за винятком того, що він заповнює спрайт, починаючи з певної точки у встановленому напрямку та кількості заданим методом.

Опція Set Native Size, доступна, коли вибрано Simple або Filled , встановлює зображення початковий розмір спрайту.

Зображення (Images) можна імпортувати як **спрайти UI** , натиснувши на Sprite(2D/UI) у параметрі 'Texture Type'. Спрайти мають додаткові параметри імпорту, порівняно зі старими спрайтами GUI, найбільша різниця – наявність редактора спрайтів. Він дає можливість **розділити зображення на 9 частин** , щоб за зміни розміру спрайту його кути не розтягувалися і спотворювалися.



Raw Image

Компонент Image приймає спрайт, а **Raw Image** – текстуру (без кордонів тощо). Raw Image краще використовувати лише тоді, коли це дійсно необхідно, в іншому Image підходить для більшості випадків.

Маска

Маска – не видимий елемент UI, а скоріше спосіб зміни зовнішнього вигляду дочірніх елеме нтів керування. Маска обмежує дочірні елементи формою батька. Таким чином, якщо дочірній елемент більший, ніж батько, то буде видно тільки ту частину, яка міститься всередині батька.

Ефекти

Visual components може також мати різні прямі ефекти застосовані, так як прямі скручування hadow або outline. Натисніть на UI Effects reference page for more information.

Компоненти взаємодії

Цей розділ присвячений компонентам в системі інтерфейсів користувача (UI), які відповідають за взаємодії, такі як події миші або дотиків, а також взаємодії з використанням клавіатури або контролера.

Компоненти взаємодії невидимі самі по собі, їх потрібно об’єднати з одним або декількома візуальними компонентами , щоб працювати правильно.

Common Functionality

Більшість компонентів взаємодії мають деякі спільні риси. Вони доступні для вибору, що означає, що вони мають спільні вбудовані функції для візуалізації переходів між станами (нормальний, виділений, натиснутий, вимкнений) і для переходу до інших вибраних за допомогою клавіатури чи контролера. Ця спільна функція описана на сторінці « Вибір » .

Компоненти взаємодії мають принаймні одну подію UnityEvent, яка викликається, коли користувач взаємодіє з компонентом певним чином. Система інтерфейсу користувача вловлює та реєструє будь-які винятки, які поширюються поза кодом, приєднаним до UnityEvent.

Button

Кнопка має **OnClick** UnityEvent, щоб визначити, що вона робитиме після натискання.

Перегляньте сторінку Button , щоб дізнатися більше про використання компонента Button.

Toggle

Перемикач має прапорець « **Увімкнено** » , який визначає, увімкнено чи вимкнено перемикач. Це значення змінюється, коли користувач натискає перемикач, і відповідно можна ввімкнути або вимкнути візуальну позначку. Він також має **OnValueChanged** UnityEvent, щоб визначити, що він робитиме, коли значення буде змінено.

Дивіться сторінку Toggle , щоб дізнатися більше про використання компонента Toggle.

Toggle Group

Групу перемикачів можна використовувати для групування набору перемикачів , які є взаємовиключними. Перемикачі, які належать до однієї групи, обмежені таким чином, що одночасно можна вибрати лише один із них – вибір одного з них автоматично скасовує вибір усіх інших.

Перегляньте сторінку « Перемикати групу» , щоб дізнатися більше про використання компонента «Перемикати групу».

Повзунок

Повзунок має десяткове числове **значення** , яке користувач може перетягувати між мінімальним і максимальним значенням. Він може бути як горизонтальним, так і вертикальним. Він також має **OnValueChanged** UnityEvent, щоб визначити, що він робитиме, коли значення буде змінено.

Перегляньте сторінку Slider , щоб дізнатися більше про використання компонента Slider.

Полоса прокрутки

Смуга прокрутки має **значення** десяткового числа від 0 до 1. Коли користувач перетягує смугу прокрутки, значення змінюється відповідно.

Смуги прокручування часто використовуються разом із Scroll Rect і Mask для створення перегляду прокручування. Смуга прокручування має значення **розміру** від 0 до 1, яке визначає розмір ручки у відношенні всієї довжини смуги прокручування. Це часто контролюється з іншого компонента, щоб вказати, наскільки велика частка вмісту в поданні прокручування є видимою. Компонент Scroll Rect може зробити це автоматично.

Смуга прокрутки може бути горизонтальною або вертикальною. Він також має **OnValueChanged** UnityEvent, щоб визначити, що він робитиме, коли значення буде змінено.

Перегляньте сторінку « Смуга прокрутки» , щоб дізнатися більше про використання компонента «Смуга прокрутки».

Dropdown

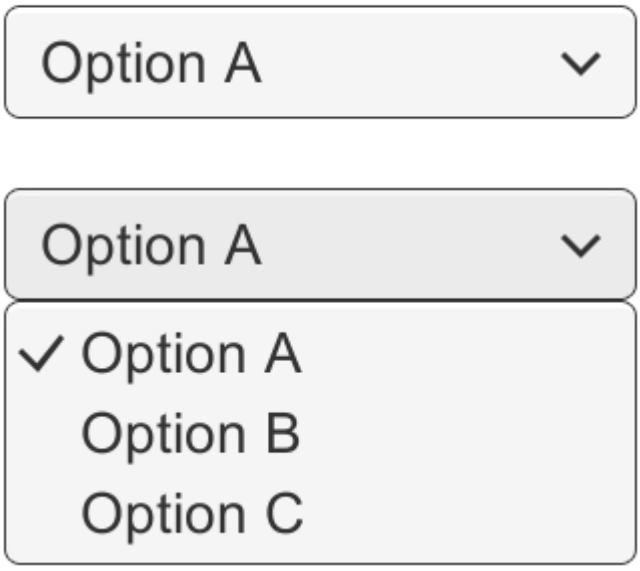
Випадаюче меню містить список опцій для вибору. Для кожного параметра можна вказати текстовий рядок і, за бажанням, зображення, які можна встановити в інспекторі або динамічно з коду. Він має **OnValueChanged** UnityEvent, щоб визначити, що він робитиме, коли поточний вибраний параметр буде змінено.

Перегляньте сторінку Dropdown , щоб дізнатися більше про використання компонента Dropdown.

Випадаюче **меню** можна використовувати, щоб дозволити користувачеві вибрати одну опцію зі списку опцій.

Елемент керування показує поточний вибраний параметр. Після натискання відкривається список параметрів, щоб можна було вибрати новий параметр. Після вибору нової опції список знову закривається, і на панелі керування відображається нова вибрана опція. Список також

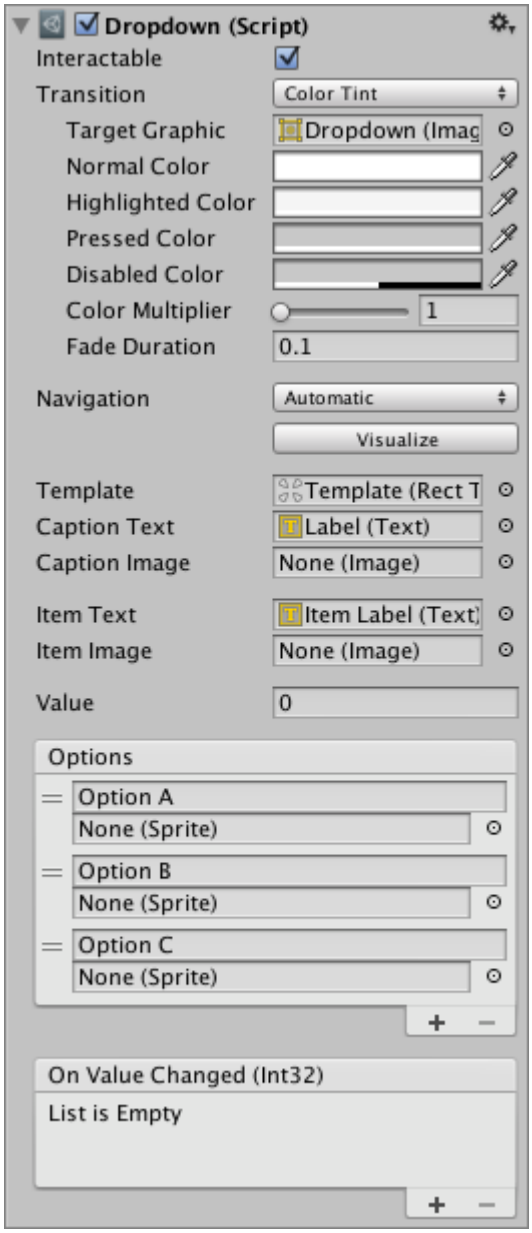
закривається, якщо користувач клацає сам елемент керування або будь-де всередині Canvas.



Випадаюче меню.

Властивість:	*функція:*
Інтерактивний	Чи прийматиме цей компонент вхідні дані? Дивіться I nteractable .
Перехід	Властивості, які визначають спосіб візуальної реакції е лемента керування на дії користувача. Перегляньте параметри переходу .
Навігація	Властивості, що визначають послідовність елементів к ерування. Див . Параметри навігації .
Шаблон	Прямое перетворення шаблону для розкривного списк у. Дивіться інструкції нижче.
Текст підпису	Компонент «Текст» для зберігання тексту поточного в ибраного параметра. (необов'язково)

Властивість:	*функція:*
Підпис зображення	Компонент «Зображення» для зберігання зображення поточного вибраного параметра. (необов'язково)
Текст елемента	Компонент Text для зберігання тексту елемента. (необ ов'язково)
Зображення предмета	Компонент Image для зберігання зображення елемента . (необов'язково)
Значення	Індекс поточного вибраного параметра. 0 – перший ва ріант, 1 – другий і так далі.
Опції	Список можливих варіантів. Для кожного параметра м ожна вказати текстовий рядок і зображення.



Розкривне меню з відкритим списком опцій.

Властивості

Події

Властивість:	*функція:*
На значення змінено	Подія UnityEvent , яка викликається, коли користувач нат искає одну з опцій у спадному списку.

Подробиці

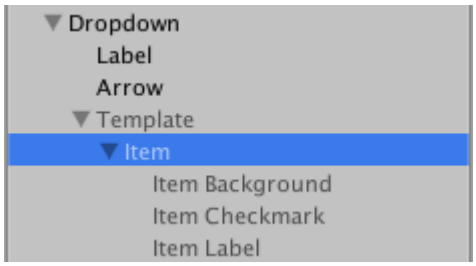
Список опцій вказується в Інспекторі або може бути призначений з коду. Для кожного параметра можна вказати текстовий рядок і, за бажанням, також зображення, якщо випадаюче меню налаштовано на підтримку цього.

Кнопка має одну подію під назвою *On Value Changed* , яка реагує, коли користувач завершує клацання однієї з опцій у списку. Він підтримує надсилання цілого числа, яке є індексом вибраного параметра. 0 – перший варіант, 1 – другий і так далі.

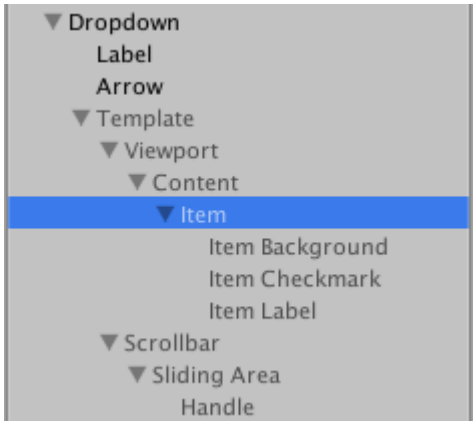
Система шаблонів

Елемент керування Dropdown розроблено таким чином, щоб мати дочірній об’єкт GameObject, який служить шаблоном для списку, що розкривається, який відображається під час натискання елемента керування спадного списку. Шаблон GameObject неактивний за умовчанням, але його можна зробити активним під час редагування шаблону, щоб краще бачити, що відбувається. Посилання на об’єкт шаблону має бути зазначено у властивості Template компонента Dropdown.

Шаблон має містити один елемент із увімкненим компонентом Toggle. Коли фактичний розкривний список створюється після клацання розкривного меню, цей елемент дублюється кілька разів, причому одна копія використовується для кожного параметра в списку. Розмір батьківського елемента автоматично змінюється, щоб він міг помістити всі елементи всередині.



Просте налаштування спадного меню, де елемент є дочірнім елементом шаблону.



Більш просунуте налаштування розкривного меню, яке включає перегляд прокрутки, який дозволяє прокручувати, коли в списку є багато параметрів.

Шаблон можна налаштувати різними способами. Параметри, які використовуються пунктом меню GameObject > UI > Dropdown, включають режим прокручування, таким чином, якщо є занадто багато параметрів для показу одночасно, з'являється смуга прокрутки, і користувач може прокручувати параметри. Однак це не є обов'язковою частиною налаштування шаблону.

(Див. сторінку ScrollRect, щоб отримати додаткові відомості про налаштування Scroll Views.)

Налаштування підтримки тексту та зображень

Розкривний список підтримує один текстовий вміст і одне зображення для кожного параметра. І текст, і зображення необов'язкові. Їх можна використовувати, лише якщо спадне меню налаштовано на підтримку.

Розкривне меню підтримує текст для кожного параметра, якщо налаштовано обидва властивості Текст підпису та Текст елемента. Вони встановлюються за замовчуванням під час використання пункту меню GameObject > UI > Dropdown.

- Текст підпису — це компонент Текст, який містить текст для поточного вибраного параметра. Зазвичай він є дочірнім для Dropdown GameObject.
- Текст елемента — це компонент Text, який містить текст для кожного параметра. Зазвичай він є дочірнім для Item GameObject.

Розкривний список підтримує зображення для кожного параметра, коли налаштовано обидва властивості Зображення підпису та Зображення елемента. Вони не налаштовані за умовчанням.

- Зображення підпису — це компонент зображення, який містить зображення для поточного вибраного параметра. Зазвичай він є дочірнім для Dropdown GameObject.
- Зображення елемента — це компонент «Зображення», який містить зображення для кожного параметра. Зазвичай він є дочірнім для Item GameObject.

Фактичний текст і зображення, які використовуються для розкривних списків, вказуються у властивості Options компонента Dropdown або можуть бути встановлені з коду.

Розміщення випадаючого списку

Розташування розкривного списку відносно елемента керування розкривним списком визначається прив'язкою та поворотом Rect Transform шаблону.

За замовчуванням список відображатиметься під елементом керування. Це досягається шляхом прикріплення шаблону до нижньої частини елемента керування. Опорна точка шаблону також має бути вгорі, щоб, коли шаблон розгортається для розміщення змінної кількості опцій, він розширювався лише донизу.

Елемент керування Dropdown має просту логіку, яка запобігає відображенню спадного списку за межами Canvas, оскільки це унеможливить вибір певних параметрів. Якщо розкривний список у положенні за замовчуванням не повністю знаходиться в межах прямокутника Canvas, його положення щодо елемента керування змінюється на протилежне. Наприклад, список, який за замовчуванням відображається під елементом керування, замість цього відображатиметься над ним.

Ця логіка досить проста і має певні обмеження. Шаблон розкривного меню має бути не більшим за половину розміру Canvas мінус розмір розкривного елемента керування, інакше може не залишитися місця для списку в будь-якій позиції, якщо розкривний елемент керування розміщено посередині Canvas.

Input Field

Поле введення використовується для того, щоб користувач міг редагувати текст текстового елемента . Він має UnityEvent, щоб визначити, що він робитиме, коли текстовий вміст буде змінено, і інший, щоб визначити, що він робитиме, коли користувач завершить його редагування.

Перегляньте сторінку « Поле введення» , щоб дізнатися більше про використання компонента «Поле введення».

Input Field

Поле **введення** — це спосіб зробити текст текстового елемента керування доступним для редагування. Як і інші елементи керування взаємодією, він сам по собі не є видимим елементом інтерфейсу і має бути поєднаний з одним або кількома елементами візуального інтерфейсу, щоб бути видимим.

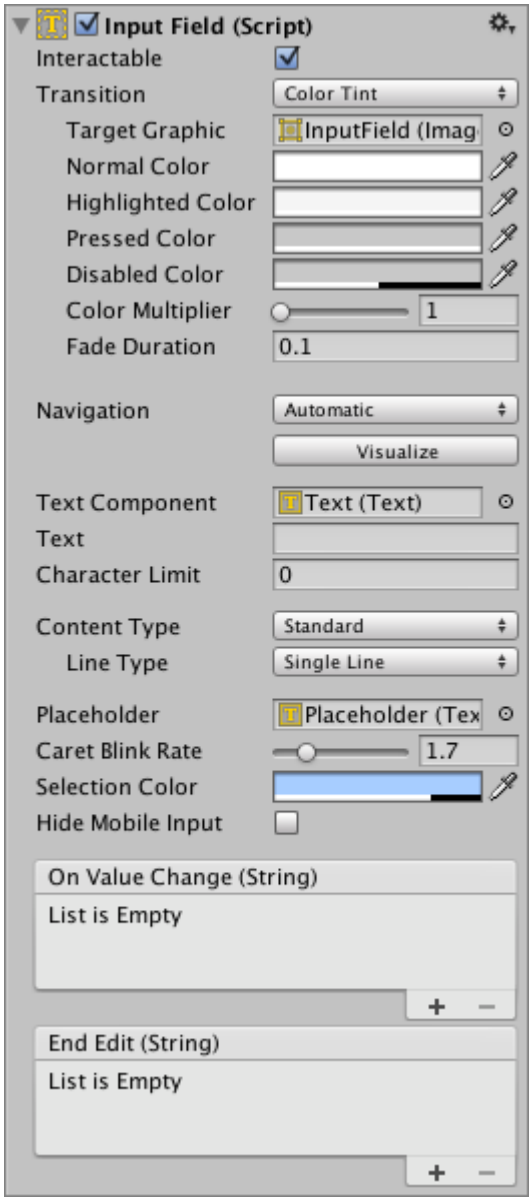
Пусте поле введення

Текст введений в поле

Властивість:	*функція:*
Інтерактивний	Логічне значення, яке визначає, чи можна взаємодіяти з полем введення.
Перехід	Переходи використовуються для встановлення способу перехо ду поля введення під час *нормального* , *виділеного* , *натиснут ого* або *вимкненого* .
Навігація	Властивості, що визначають послідовність елементів керуванн я. Див . Параметри навігації .
TextComponent	Посилання на елемент Text , який використовується як вміст п оля *введення*
текст	Початкова вартість. Початковий текст, розміщений у полі пере д початком редагування.
Обмеження символів	Значення максимальної кількості символів, які можна ввести в поле введення.
Тип вмісту	Визначте тип(и) символів, які приймає ваше поле введення
Стандартни й	Можна ввести будь-який символ.
Автовиправ лено	Автовиправлення визначає, чи введені дані відстежують невідомі слова, і пропонує користувачеві більш прийнятний варіант заміни, автоматично замінюючи введений текст, якщо користувач явно не скасовує дію.
Ціле число	Дозволити вводити лише цілі числа.
Десяткове ч исло	Дозволяє вводити лише числа та одну десяткову крапку.

Властивість:	*функція:*
буквено-цифровий	Дозволяють як літери, так і цифри. Не можна вводити символи.
Ім'я	Автоматично робить першу літеру кожного слова великою. Зверніть увагу, що користувач може обійти правила використання великих літер за допомогою клавіші **Delete** .
Адреса електронної пошти	Дозволяє ввести буквено-цифровий рядок, що складається максимум з одного знака @. крапки/крапки базової лінії не можна вводити одна біля одної.
пароль*	Приховує символи, введені із зірочкою. Дозволяє символи.
Pin	Приховує символи, введені із зірочкою. Дозволяє вводити лише цілі числа.
Custom	Дозволяє налаштувати тип рядка, тип введення, тип клавіатури та перевірку символів.
Тип лінії	Визначає форматування тексту в текстовому полі.
Одна лінія	Дозволяє текст лише в один рядок.
Багаторядкове надсилання	Дозволяє тексту використовувати кілька рядків. Використовує новий рядок лише за потреби.
Багаторядковий новий рядок	Дозволяє тексту використовувати кілька рядків. Користувач може використовувати новий рядок, натиснувши клавішу повернення.
Заповнювач	Це необов’язкова «порожня» графіка , яка показує, що в полі *введення* немає тексту. Зауважте, що ця «порожня» графіка все

Властивість:	*функція:*
	ще відображається, навіть якщо вибрано *поле введення* (тобто, коли на ньому знаходиться фокус). наприклад; «Введіть текст …».
Частота мигання курсору	Визначає частоту блимання для позначки, розміщеної на рядку для позначення пропонованої вставки тексту.
Колір вибору	Колір фону виділеної частини тексту.
Приховати мобільний вхід (тільки для iOS)	Приховує рідне поле введення, приєднане до екранної клавіату ри на мобільних пристроях. Зауважте, що це працює лише на п ристроях iOS.


Input Field (Script)
Interactable
Transition
Color Tint
Target Graphic
InputField (Imag
Normal Color
Highlighted Color
Pressed Color
Disabled Color
Color Multiplier
1
Fade Duration
0.1
Navigation
Automatic
Visualize
Text Component
Text (Text)
Text
Character Limit
0
Content Type
Standard
Line Type
Single Line
Placeholder
Placeholder (Tex
Caret Blink Rate
1.7
Selection Color
Hide Mobile Input
On Value Change (String)
List is Empty
End Edit (String)
List is Empty

Властивості

Події

Властивість:	*функція:*
Про зміну вартості	Подія UnityEvent , яка викликається, коли змінюється текстовий вміст поля введення. Подія може надсилати поточний текстовий вміст як string динамічний аргумент типу.
Завершити редагування	Подія UnityEvent , яка викликається, коли користувач завершує редагування текстового вмісту шляхом надсилання або клацання десь, що знімає фокус із поля введення. Подія може надсилати поточний текстовий вміст як string динамічний аргумент типу.

Подробиці

Сценарій поля введення можна додати до будь-якого існуючого об’єкта керування текстом за допомогою меню (**Компонент > Інтерфейс користувача > Поле введення**). Зробивши це, ви також повинні перетягнути об’єкт до властивості *Текст* поля введення, щоб увімкнути редагування.

Властивість *Text* самого елемента керування Text змінюватиметься, коли користувач вводить текст, і значення можна буде отримати зі сценарію після редагування. Зауважте, що форматований текст навмисно не підтримується для редагованих елементів керування текстом; у цьому полі миттєво буде застосовано будь-яку розмітку форматованого тексту під час введення, але розмітка фактично «зникає», і подальшого способу змінити або видалити стиль неможливо.

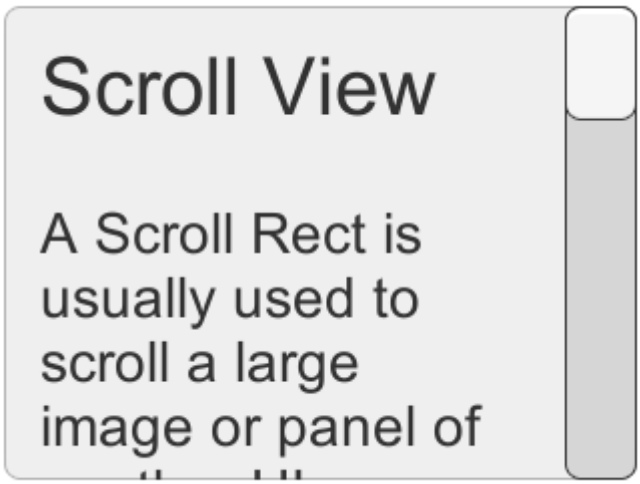
Поради

- Щоб отримати текст поля введення, використовуйте властивість text самого компонента InputField, а не властивість text компонента Text, який відображає текст. Властивість text компонента Text може бути обрізана або може складатися із зірочок для паролів.

Scroll Rect (перегляд прокрутки)

Scroll Rect можна використовувати, коли вміст, який займає багато місця, потрібно відобразити в невеликій області. Scroll Rect забезпечує функціональність для прокручування цього вмісту.

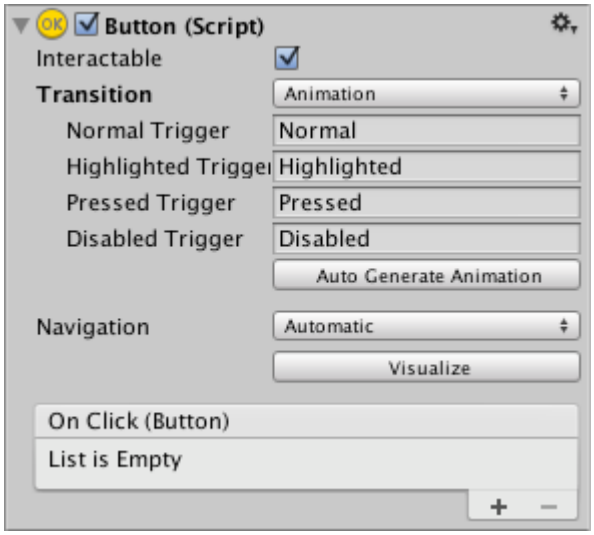
Зазвичай прямокутник прокручування поєднується з маскою для створення перегляду прокручування, де видно лише вміст, який можна прокручувати всередині прямокутника прокручування. Його також можна додатково поєднати з однією або двома смугами прокручування , які можна перетягувати для прокручування горизонтально або вертикально.



Дивіться сторінку Scroll Rect , щоб дізнатися більше про використання компонента Scroll Rect.

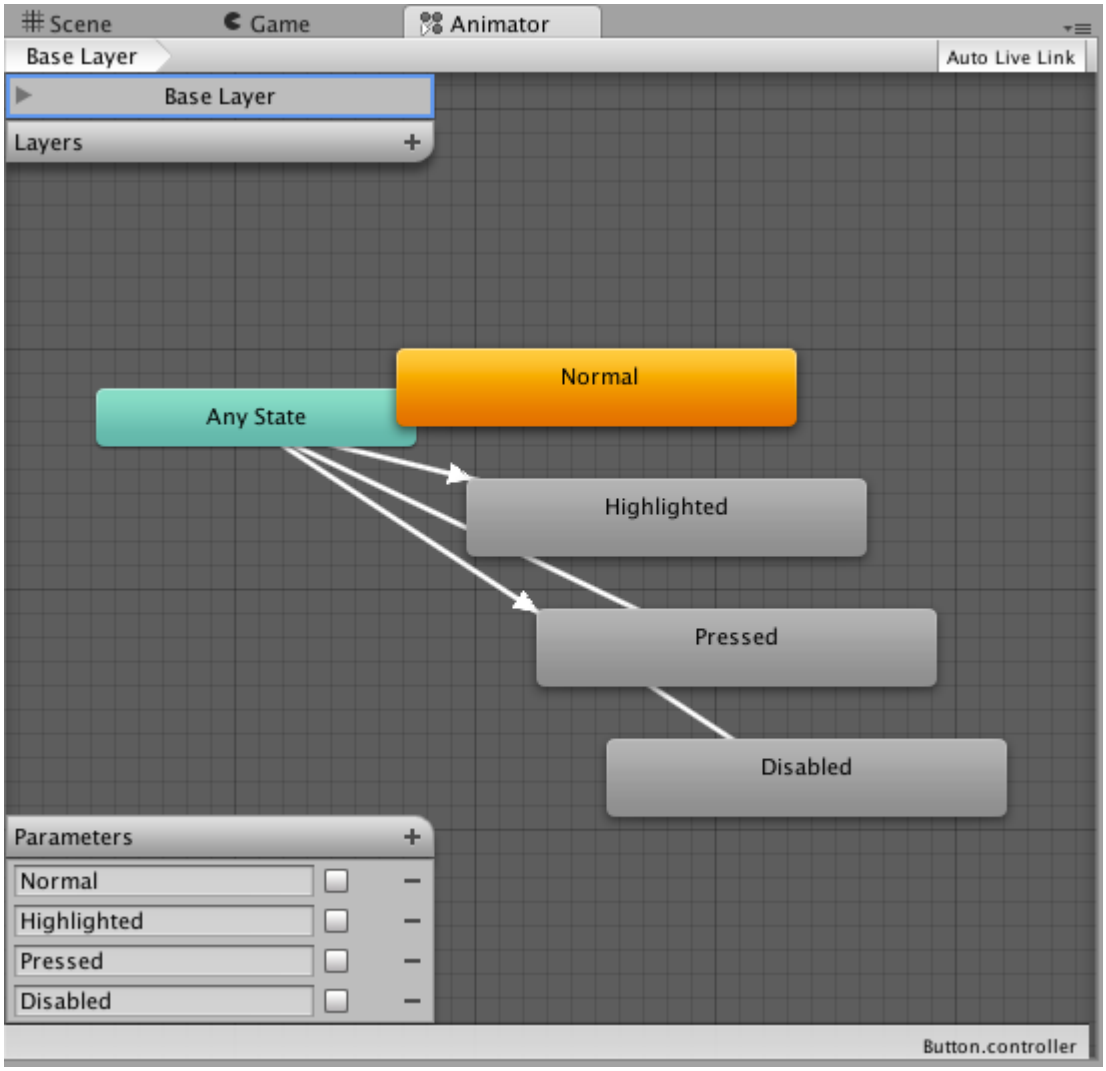
Animation Integration

Анімація дозволяє повністю анімувати кожен перехід між станами керування за допомогою системи анімації Unity. Це найпотужніший із режимів переходу завдяки кількості властивостей, які можна анімувати одночасно.



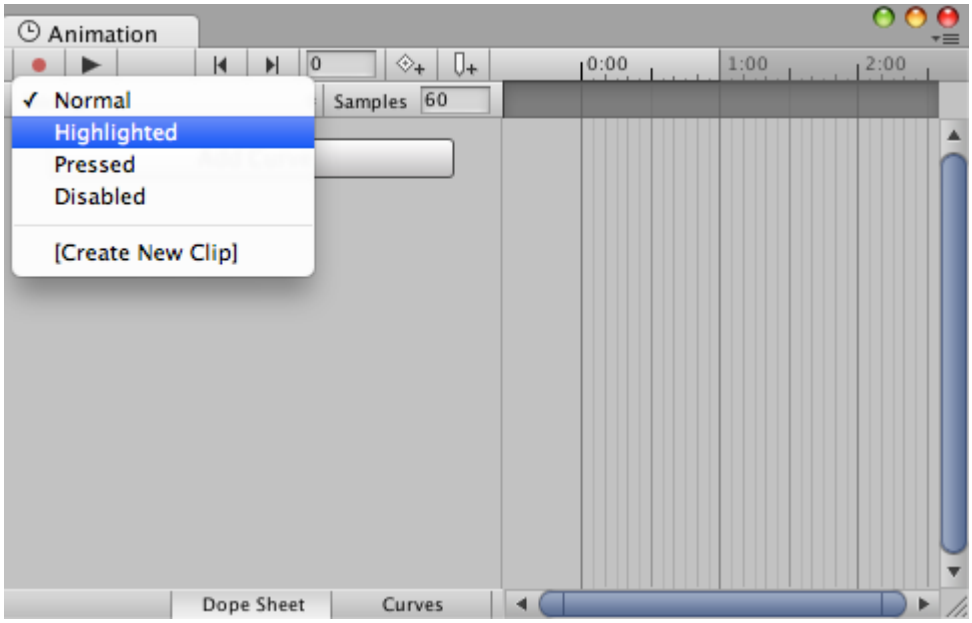
Щоб використовувати режим переходу Animation, компонент Animator потрібно прикріпити до елемента контролера. Це можна зробити автоматично, натиснувши «Автоматичне створення анімації». Це також генерує контролер аніматора з уже налаштованими станами, які потрібно буде зберегти.

Новий контролер Animator готовий до використання одразу. На відміну від більшості контролерів Animator, цей контролер також зберігає анімації для переходів контролера, і їх можна налаштувати за бажанням.



Наприклад, якщо вибрано елемент Button із приєднаним контролером Animator, анімацію для кожного стану кнопки можна редагувати, відкривши вікно Animation (**Window>Animation**).

Існує спливаюче меню Animation Clip для вибору потрібного кліпу. Виберіть «Звичайний», «Виділений», «Натиснутий» або «Вимкнений».



Нормальний стан встановлюється значеннями на самому елементі кнопки, і його можна залишити порожнім. У всіх інших станах найпоширенішою конфігурацією є один ключовий кадр на початку шкали часу. Анімація переходу між станами оброблятиметься аніматором.

Як приклад, ширину кнопки у виділеному стані можна змінити, вибравши виділений стан у спливаючому меню анімаційного кліпу та вказавши вказівник відтворення на початку шкали часу:

- Увімкніть режим запису
- Змініть ширину кнопок в інспекторі
- Вимкніть режим запису.
  Тепер, при наведенні курсору на кнопку в ігровому режимі, вона зростатиме.

Будь-яка кількість властивостей може мати свої параметри в одному ключовому кадрі.

Кілька кнопок можуть мати однакову поведінку за допомогою спільних контролерів Animator.

Практичне завдання

Кожному студенту буде видане індивідуальне завдання. Використовуючи Unity 3D та інші необхідні інструменти, потрібно розробити працюючий додаток - гру.

Вимоги до гри:

- Потрібно зробити мінімум 3 сцени (основна гра, правила гри, про розробника.
- Якщо ви берете існуючу гру (наприклад хрестику-нолики) то необхідно додати щось своє. Це може бути інший спосіб підрахунку балів, або обов'язкова мінімальна серія ігор і т.д. Необхідно проявити креативність.
- Доповнити гру музичним супроводом. Як мінімум це може бути звук натискування на кнопку, а як максимум фонова музика.

Номер варіанту відповідає порядковому номеру студента в списку групи. Студент може запропонувати свою власну гру, але вона не повинна співпадати з любим з існуючих варіантів завдання.

В написанні звіту детально описати алгоритм гри. Додати лістинги скриптів на мові C# та скрін-шоти роботи програми.

Сам проєкт та звіт необхідно завантажити в Moodle.

№ варіанту	Завдання
1	Розробити графічний інтерфейс гри Хрестики та Нулики. Задіяти розмір 3x3 Клітини. Почергово гравці роблять хід та мають заповнити рядок, стовбець, рядок чи діагональ однаково знаками. Виграє той хто перший це зробить. Додатково показувати чий хід зараз.. Зберігати результати гри.
2	Розробити спрощений варіант гри «однорукий бандит». Зробити кнопки «Start», можливість встановити ставку у грі.

№ варіанту	Завдання
	Після натискання кнопки старт у трьох віконцях по черзі випадкове чином встановлюються числа від 1 до 7 включно. В залежності від кількості однаково чисел, встановлюється сума виграшу, яка додається до коштів гравця. Закінчується гра при закінченні коштів у гравця.
3	Розробити спрощений варіант гри «Сапер». Поле 9Х9 клітини. Спочатку гри всі кнопки не мають надписів. При натискуванні на кнопку і якщо не має міни то на кнопці появляється надпис з кількістю мін жовтим на блакитному фоні. Якщо попадає на міну, то кнопка зафарбовується червоним и гра припиняється. З початку першого ходу включається таймер. Додатково є кнопка початку нової гри.
4	Розробити спрощений варіант гри «однорукий бандит». Зробити кнопки «Start», можливість встановити ставку у грі. Після натискання кнопки старт у трьох віконцях по черзі випадкове чином встановлюються числа від 1 до 7 включно. В залежності від кількості однаково чисел, встановлюється сума виграшу, яка додається до коштів гравця. Закінчується гра при закінченні коштів у гравця.
5	Розробити гру Memory. Суть гри в тому що вам пропонують 8 картинок в перевернутому вигляді. 4 пари однакових картинок. Ви по черзі відкриваєте дві картинки і якщо вони співпадають то картинки зникають. Потрібно найменшою кількістю рухів очистити ігрове поле. Вести кількість відкривання карт та запропонувати систему підрахунку балів.
6	Розробити гру 10. Формується поле 9Х9 клітинок. В кожному рядку довільним чином розташовуються числа від 1 до 9. Для початку гри програма випадковим чином вибирає клітинку та знищує всі клітинки довкола цієї точки. Таким чином створюються вільний простір. Вам потрібно вибрати дві цифри сума яких би давала 10. Перша цифра просто зникає а остання цифра у вашому вибору вибухає та знищує цифри які дотичні до неї. Зробити систему підрахунку балів.

№ варіанту	Завдання
7	Розробити гру міні-судоку. Розмір сітки 4х4, цифри від 1 до 4 в кожному рядку. Зробити систему розрахунку балів. При невірному натисненню додавати штрафні бали. Мінімальна складність.
8	Розробити спрощений варіант гри «однорукий бандит». Зробити кнопки «Start», можливість встановити ставку у грі. Після натискання кнопки старт у трьох віконцях по черзі випадкове чином встановлюються числа від 1 до 7 включно. В залежності від кількості однаково чисел, встановлюється сума виграшу, яка додається до коштів гравця. Закінчується гра при закінченні коштів у гравця. У гравця є можливість закінчити гру в любий момент часу та зберегти кількість набраних очок.
9	Розробити спрощений варіант гри «однорукий бандит». Зробити кнопки «Start», можливість встановити ставку у грі. Після натискання кнопки старт у трьох віконцях по черзі випадкове чином встановлюються числа від 1 до 7 включно. В залежності від кількості однаково чисел, встановлюється сума виграшу, яка додається до коштів гравця. Закінчується гра при закінченні коштів у гравця. Тло вікна зробити зеленим кольором, випадкові числа зробити червоним на чорному фоні. Зробити можливим запис та зберігання очок для кожного гравця. При ініціалізації гри можна буде вибрати існуючого гравця, або добавити нового
10	Розробити гру 10. Формується поле 12Х12 клітинок. В кожному рядку довільним чином розташовуються числа від 1 до 9. Для початку гри програма випадковим чином вибирає клітинку та знищує всі клітинки довкола цієї точки. Таким чином створюються вільний простір. Вам потрібно вибрати дві цифри сума яких би давала 10. Перша цифра просто зникає а остання цифра у вашому вибору вибухає та знищує цифри які дотичні до неї. Зробити систему підрахунку балів.
11	Розробити графічний інтерфейс гри Хрестики та Нулики. Задіяти розмір 3х3 Клітини. Почергово гравці роблять хід та

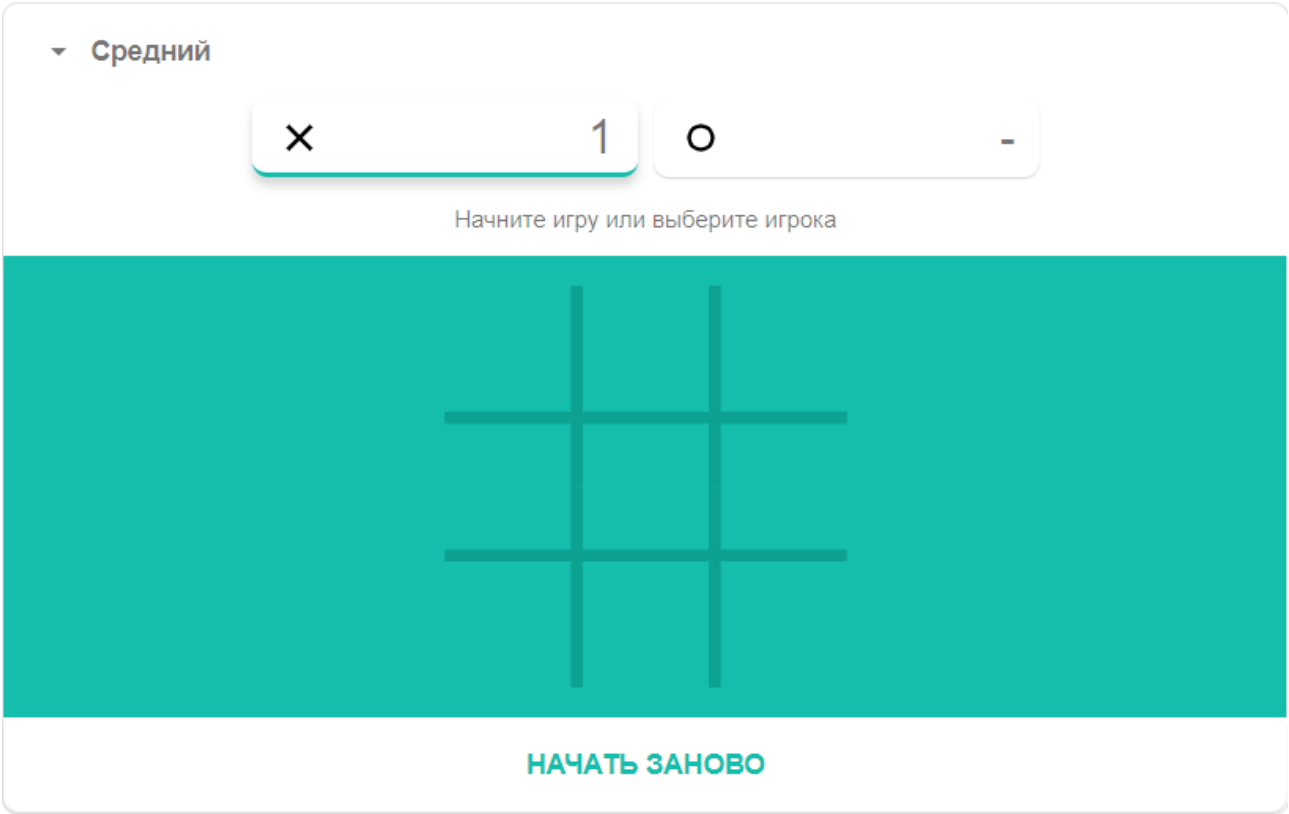
№ варіанту	Завдання
	мають заповнити рядок, стовбець, рядок чи діагональ однаково знаками. Виграє той хто перший це зробить. Додатково показувати чий хід зараз.. Зберігати результати гри.
12	Розробити спрощений варіант гри «Сапер». Поле 9Х9 клітини. Спочатку гри всі кнопки не мають надписів. При натискуванні на кнопку і якщо не має міни то на кнопці появляється надпис з кількістю мін жовтим на блакитному фоні. Якщо попадає на міну, то кнопка зафарбовується червоним и гра припиняється. З початку першого ходу включається таймер. Додатково є кнопка початку нової гри.
13	Розробити спрощений варіант гри «однорукий бандит». Зробити кнопки «Start», можливість встановити ставку у грі. Після натискання кнопки старт у трьох віконцях по черзі випадкове чином встановлюються числа від 1 до 7 включно. В залежності від кількості однаково чисел, встановлюється сума виграшу, яка додається до коштів гравця. Закінчується гра при закінченні коштів у гравця. Тло вікна зробити малюнком грального стола в казино, випадкові числа зробити жовтими на червоному фоні.
14	Розробити гру розведення машинок з майданчика. На парковці знаходиться 4 машини. Розробіть завдання згідно лабораторних № 4,5,6.Додатково добавте можливість задавати рух машинки вперед чи назад. Враховуйте можливе зіткнення машинок та при зіткненні додайте ефект. Рахуйте кількість кліків та машинок що залишилися.
15	Розробити графічний інтерфейс гри Хрестики та Нулики. Задіяти розмір 3х3 Клітини. Почергово гравці роблять хід та мають заповнити рядок, стовбець, рядок чи діагональ однаково знаками. Виграє той хто перший це зробить. Додатково показувати чий хід зараз. Зберігати результати гри.
16	Розробити спрощений варіант гри «однорукий бандит». Зробити кнопки «Start», можливість встановити ставку у грі. Після натискання кнопки старт у трьох віконцях по черзі

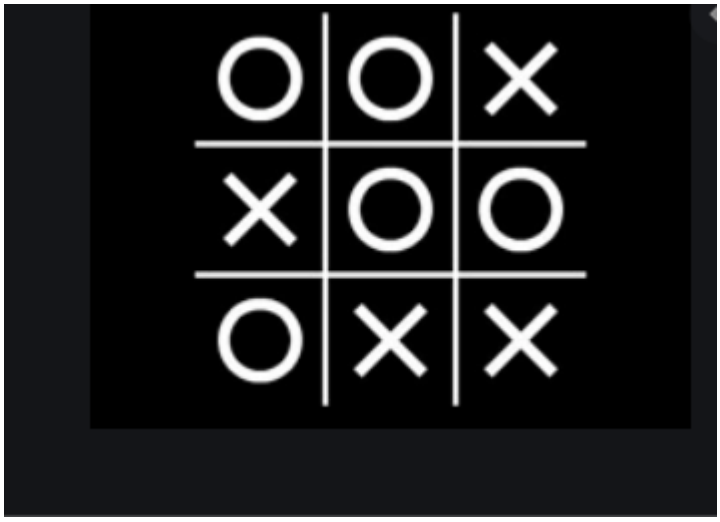
№ варіанту	Завдання
	випадкове чином встановлюються числа від 1 до 7 включно. В залежності від кількості однаково чисел, встановлюється сума виграшу, яка додається до коштів гравця. Закінчується гра при закінченні коштів у гравця.
17	Розробити спрощений варіант гри «Сапер». Поле 9Х9 клітини. Спочатку гри всі кнопки не мають надписів. При натискуванні на кнопку і якщо не має міни то на кнопці появляється надпис з кількістю мін жовтим на блакитному фоні. Якщо попадає на міну, то кнопка зафарбовується червоним и гра припиняється. З початку першого ходу включається таймер. Додатково є кнопка початку нової гри.
18	Розробити спрощений варіант гри «однорукий бандит». Зробити кнопки «Start», можливість встановити ставку у грі. Після натискання кнопки старт у трьох віконцях по черзі випадкове чином встановлюються числа від 1 до 7 включно. В залежності від кількості однаково чисел, встановлюється сума виграшу, яка додається до коштів гравця. Закінчується гра при закінченні коштів у гравця.
19	Розробити гру Memory. Суть гри в тому що вам пропонують 8 картинок в перевернутому вигляді. 4 пари однакових картинок. Ви по черзі відкриваєте дві картинки і якщо вони співпадають то картинки зникають. Потрібно найменшою кількістю рухів очистити ігрове поле. Вести кількість відкривання карт та запропонувати систему підрахунку балів.
20	Розробити гру 10. Формується поле 9Х9 клітинок. В кожному рядку довільним чином розташовуються числа від 1 до 9. Для початку гри програма випадковим чином вибирає клітинку та знищує всі клітинки довкола цієї точки. Таким чином створюються вільний простір. Вам потрібно вибрати дві цифри сума яких би давала 10. Перша цифра просто зникає а остання цифра у вашому вибору вибухає та знищує цифри які дотичні до неї. Зробити систему підрахунку балів.

№ варіанту	Завдання
21	Розробити гру міні-судоку. Розмір сітки 4х4, цифри від 1 до 4 в кожному рядку. Зробити систему розрахунку балів. При невірному натисненню додавати штрафні бали. Мінімальна складність.
22	Розробити спрощений варіант гри «однорукий бандит». Зробити кнопки «Start», можливість встановити ставку у грі. Після натискання кнопки старт у трьох віконцях по черзі випадкове чином встановлюються числа від 1 до 7 включно. В залежності від кількості однаково чисел, встановлюється сума виграшу, яка додається до коштів гравця. Закінчується гра при закінченні коштів у гравця. У гравця є можливість закінчити гру в любий момент часу та зберегти кількість набраних очок.
23	Розробити спрощений варіант гри «однорукий бандит». Зробити кнопки «Start», можливість встановити ставку у грі. Після натискання кнопки старт у трьох віконцях по черзі випадкове чином встановлюються числа від 1 до 7 включно. В залежності від кількості однаково чисел, встановлюється сума виграшу, яка додається до коштів гравця. Закінчується гра при закінченні коштів у гравця. Тло вікна зробити зеленим кольором, випадкові числа зробити червоним на чорному фоні. Зробити можливим запис та зберігання очок для кожного гравця. При ініціалізації гри можна буде вибрати існуючого гравця, або добавити нового
24	Розробити гру 10. Формується поле 12Х12 клітинок. В кожному рядку довільним чином розташовуються числа від 1 до 9. Для початку гри програма випадковим чином вибирає клітинку та знищує всі клітинки довкола цієї точки. Таким чином створюються вільний простір. Вам потрібно вибрати дві цифри сума яких би давала 10. Перша цифра просто зникає а остання цифра у вашому вибору вибухає та знищує цифри які дотичні до неї. Зробити систему підрахунку балів.
25	Розробити графічний інтерфейс гри Хрестики та Нулики. Задіяти розмір 3х3 Клітини. Почергово гравці роблять хід та

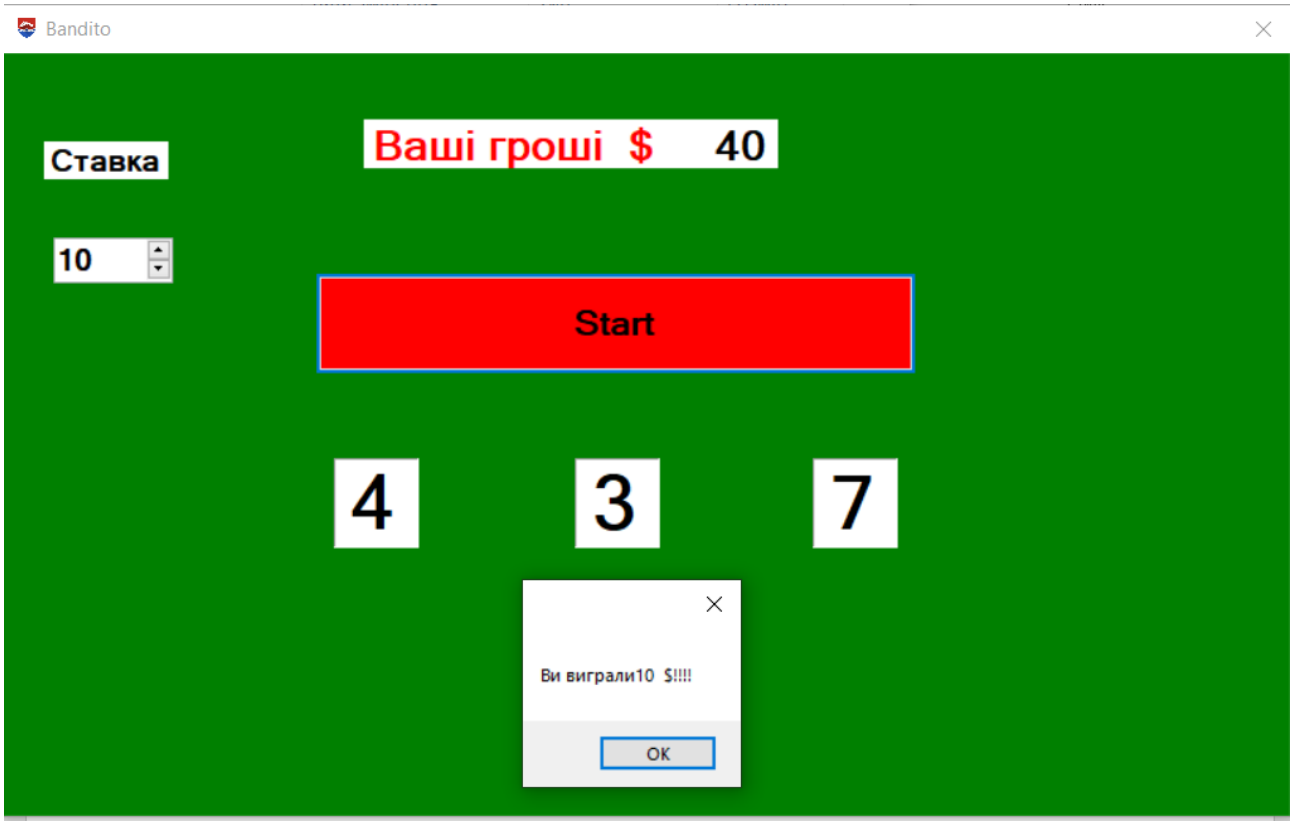
№ варіанту	Завдання
	мають заповнити рядок, стовбець, рядок чи діагональ однаково знаками. Виграє той хто перший це зробить. Додатково показувати чий хід зараз.. Зберігати результати гри.

Приклад вигляд гри хрестики та нулики:





Приклад спрощений варіанта гри «однорукий бандит»:



Роботу оформити згідно стандарту. На титульній сторінці вказати Міністерство та назва університету, Зазначити виконавця, групу, рік. Приклад титульної Сторінки наведено нижче:

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ МОРСЬКИЙ УНІВЕРСИТЕТ

Кафедра «Технічної кібернетики й інформаційних технологій ім.

проф. Р.В. Меркта »

Розрахунково ГРАФІЧНЕ ЗАВДВННЯ

з дисципліни «Основи розробки ігор»

на тему: «Розробка гри мінер»

Варіант № 3

Виконаю: Студент (ка) _2_ курсу _2_ групи

спеціальності 122 «Комп'ютерні науки та інформаційні технології ».

_____

(Прізвище та ініціали)

Перевірив:

_____

_____

(Посада, Вчене звання, науковий ступень, прізвище та ініціали)

Національна шкала _____

Кількість балів: _____ Оцінка: ECTS _____

_____ _____

(Підпис) (прізвище та ініціали)

_____ _____

(Підпис) (прізвище та ініціали)

Одеса - 2025

В подальшому описати алгоритм роботи програми та основні використані функції, методи та класи. Роздрукувати лістинг програми та скрін-шоти роботи програми. Звіт можна подавати як в паперовій так и в без паперовій формі: файли в форматі * .docx, * .odt та * .pdf.

Рекомендована література

Базова

1. Проектування і створення ігрових додатків : навчальний посібник для студентів

спеціальностей 122 "Комп'ютерні науки" та 151 "Автоматизація та комп'ютерно-інтегровані технології", спеціалізація "Системна інженерія" / І.В. Гребеннік, Є.В. Губаренко, О.В. Хряпкін ; Міністерство освіти і науки України, Харківський національний університет радіоелектроніки. - Харків : ХНУРЕ, 2018. - 116 с.

https://opac.kpi.ua/F/TGBM1YMEAMP6IG62PPLJB5C8I4KHMAMGRS9N597H4GFGXX3RCX-

04524?func=full-set-set&set_number=008290&set_entry=000002&format=999

2. Software production and game modeling methods : collective monograph / V.B. Kyselov, V.I. Domnich, M.H. Medvediev, O.M. Muliava ; V.I. Vernadsky Taurida National University. - Lviv :Toruń : Liha-Pres, 2019. - 176 с.

https://opac.kpi.ua/F/TGBM1YMEAMP6IG62PPLJB5C8I4KHMAMGRS9N597H4GFGXX3RCX-04402?func=full-set-et&set_number=008290&set_entry=000001&format=999

3. Jeremy Gibson Bond Introduction to Game Design, Prototyping, and Development: From Concept to Playable Game with Unity and C# 3rd Edition, Kindle Edition. ISBN-13 - 978-0136619949, 2022, 1000p

4. Joe Hocking Unity in Action, Third Edition: Multiplatform game development in C#. ISBN-978-1617299339, 2022.

Допоміжна

1.Проектування комп’ютерних ігор для навчання: навчальний підручник /Т.А. Лугова, О.А. Блажко. – Одеса : ФОП «Побута». – 2018. – 212 с.

2. Технології розробки комп’ютерних ігор: довідник модуля. / В.С. Бреславець.- Х.: «ДрукарняМадрид», 2018. - 162 с.

3. Основи розробки комп’ютерних ігор: електронний навчальний посібник для підготовки студентів на першому (бакалаврському) рівні вищої освіти, галузі знань 12 «Інформаційнітехнології», спеціальності 121 «Інженерія програмного забезпечення» / Укладач: В.Г.Шерстюк. – Херсон: видавництво ФОП Вишемирський В.С., 2018. – 210 с.

4. https://onmu-moodle.od.ua/course/view.php?id=1847