Рівень 6

Шалений Фелікс

Передмова: У цьому проекті ви будете створювати доволі складну гру. Це ігрова платформа, по якій рухається кіт Фелікс, ухиляючись від ворогів та перешкод і збираючи ключі, щоб втекти з печери. Коли йому це вдається, він переміщається на наступний рівень, на якому він теж повинен втекти з печери.

У перший тиждень роботи над проектом ви опануєте скрипти, що відповідають за рух Фелікса та його взаємодію з різними предметами. Впродовж другого тижня ви займатиметися створенням різних рівнів складності для гри. А на третій тиждень ви поєднаєте всі рівні у цілісну гру.

Тиждень 1: Створення блоків

Ігрові платформи такого типу, як Manic Miner і Mario Bros, побудовані таким чином, що фігурка головного героя рухається в різних світах, натикаючись на певні предмети. Деякі предмети, наприклад, стіни і перекриття, просто зупиняють рух героя. Інші ж – негативні персонажі - вбивають головного героя, якщо він на них натикається. Є також предмети, які треба збирати, наприклад, ключі. Деякі предмети просто формують тло і не впливають на гру взагалі.

Це означає, що важливо визначити момент, коли головний герой натикається на певний предмет (це називається "фіксація моменту зіткнення"). У Скретч існує кілька блоків команд, за допомогою яких можна фіксувати момент зіткнення: (`доторкається [спрайт] ?; доторкається кольору [] ?; колір [] доторкається [] ?; відстань до [спрайт]?).` Але для того, щоб створити гру, потрібно не лише знати про сам факт зіткнення героя з певним предметом, а й про те, якою саме стороною він наткнувся на нього. Якщо герой наткнувся на ліву стіну, він не можете далі рухатись ліворуч. Але він ще можете рухатись праворуч, стрибати чи падати вниз, якщо немає перекриття.

Зіткнення з ворогом може вбити головного героя, але якщо головний герой торкнеться ворога ногами, то ліквідує його. Жоден з вбудованих блоків Скретч не дає змоги виявити місця та напрямку зіткнення. Для цього треба створити власний детектор зіткнення.

Завдання 1: Детектори зіткнення

Ми створимо 4 спрайти, які супроводжуватимуть Несамовитого Фелікса. Кожен з них фіксуватиме зіткнення в одному з напрямків і задаватиме значення змінної у разі настання зіткнення. Головний спрайт Фелікс використовуватиме ці змінні для контролю

за рухом героя. Кожен з детекторів зіткнень матиме колір та міститиме блок "`колір [] доторкається [] ?`" для визначення предметів. Перекриття описуватиме чорний колір, а перешкоди – зелений. Червоним кольором позначимо детектори зіткнення.

Перелік завдань до виконання

1. **Почніть новий проект**. Видаліть звичайне біле тло і додайте на сцену тло the **frantic-felix/testlevel** (несамовитий фелікс/тестовий рівень)
2. **Переіменуйте Спрайт1 на Фелікс**. Встановіть для нього стиль обертання ліворуч-праворуч.
3. Створіть 4 нових спрайти із зображень **frantic-felix/top** (несамовитий фелікс/верх), **frantic-felix/bottom** (несамовитий фелікс/низ), **frantic-felix/left** (несамовитий фелікс/лівий), **frantic-felix/right** (несамовитий фелікс/правий). Ці спрайти будуть детекторами зіткнень. Назвіть нові спрайти Верх, Низ, Лівий, Правий, подбайте, щоб вони не перевертались і не крутились під час руху.
4. Створіть 4 змінних (для всіх спрайтів): блокувати верх, блокувати низ, блокувати праворуч, блокувати ліворуч.
5. Для кожного спрайта–детектора пропишіть такий скрипт:

```
коли натиснуто зелений прапорець
іти до Фелікс
якщо ‹колір [] доторкається [] ?› то
встановити блокувати праворуч в 1
інакше
встановити блокувати праворуч в 0
```

6. Змініть змінні для кожного детектора. Для детектора Низ слід прописати блок команд "або", щоб змінна "блокувати низ" встановлювалась при зіткненні і з чорним, і з зеленим кольором. Підказка: простіше буде вибрати кольори в одному спрайті, а потім перетягнути скрипт для інших спрайів і змінити в ньому змінні, які повинні встановлюватись. Так вам не доведеться шукати правильні кольори 4 рази. Залишилось тільки створити скрипт, який прописуватиме рух Фелікса за вказівником миші завжди.

Протестуйте свій проект

Натисніть на зелений прапорець.

Ви повинні побачити, як Фелікс слідує за вказівником миші на тлі червоного прямокутника. Прямокутник утворюють детектори зіткнення. Якщо ви слідкуватиме за змінними, то побачите що вони змінюються, коли ви переміщуєте Фелікса по екрану і він торкається різних його частин. Поки що Фелікс рухається крізь платформу та зелені перешкоди. Виконуючи наступні завдання, ви усунете ці недоліки.

Збережіть свій проект.

Завдання 2: Рух, рівні й падіння

1. Час подумати про те, як Фелікс рухатиметься. Керувати його рухом ми будемо за допомогою клавіш ***стрілка ліворуч*** і ***стрілка праворуч***. Якщо Фелікс не стоїть на твердій поверхні (перекриті), він падатиме вниз. (Стрибки ми опишемо далі.) При натисканні клавіші стрілка ліворуч Фелікс повертатиметься ліворуч і проходитиме деяку відстань. Нам також важливо, щоб він не рухався далі ліворуч після того, як зіткнеться з якоюсь перешкодою.
2. Можна використати команду «коли натиснуто клавішу []» для того, щоб прописати скрипт руху Фелікса, але тоді його рух буде уривчастим. Краще скористатись циклом «завжди» і командою «якщо клавішу стрілка ліворуч натиснуто? то». Для спрацювання детектора зіткнення потрібно прописати у циклі «якщо» тестування змінної «блокувати ліворуч» за допомогою команди «і», щоб Фелікс рухався ліворуч тоді, коли натиснуто стрілку ліворуч, а змінній «блокувати ліворуч» присвоєно значення 0. Аналогічно треба прописати рух Фелікса праворуч.
3. Можна прописати анімацію для ніг Фелікса у цьому ж блоці команд, але тоді вони будуть рухатись занадто швидко. Краще зробити це в окремому циклі «завжди», який виконуватиметься, коли натиснуто зелений прапорець.
4. Залишилось попрацювати ще з падінням Фелікса. Якщо Фелікс не торкається ногами якоїсь поверхні, він повинен падати. Для цього потрібно прописати команди в циклі «завжди», які виконуватимуться при натисканні зеленого прапорця.

```
коли натиснуто зелений прапорець (продовження руху)
завжди
 якщо <<клавішу стрілка ліворуч натиснуто?> і (блокувати ліворуч)=0> то
 повернути в напрямку -90
 переміститись на 2 кроків
 кінець якщо
 якщо <<клавішу стрілка праворуч натиснуто?> і (блокуваи праворуч)=0> то
 повернути в напрямку -90
 переміститись на 2 кроків
 кінець якщо
кінець завжди


коли натиснуто зелений прапорець (анімація Фелікса)
завжди якщо << клавішу стрілка ліворуч натиснуто?>або< клавішу стрілка праворуч натиснуто?>>
 наступний образ
 чекати 0.1 секунд
кінець якщо

коли натиснуто зелений прапорець (продовження падіння)
завжди
```

```
 якщо <блокувати низ=0>
 змінити на -2
 кінець якщо
кінець завжди
```

Протестуйте свій проект

За допомогою мишки перетягніть Фелікса у певне місце сцени і натисніть зелений прапорець. Якщо Фелікс підстрибне до курсора, видаліть скрипт! Фелікс повинен рухатись з боку в бік при натисканні клавіш стрілка ліворучі стрілка праворуч. Якщо він не стоїть на перекриттті, він повинен повільно падати.

1. Детектори зіткнень не повинні відображатись. Зробити це за допомогою блоку команди «сховати» не можна, бо тоді вони не фіксуватимуть зіткнень. Натомість пропишіть команду встановити ефект привид в 100, який буде виконуватись одразу після того, коли натиснуто зелений прапорець для кожного з детекторів зіткнень. Так спрайт можна зробити невидимим, не ховаючи його.

Протестуйте свій проект

При натисканні зеленого прапорця детектори зіткнень повинні стати невидимими. Вони знову з’являться при натисканні червоної кнопки стоп.

Збережіть свій проект

Завдання 3: Стрибки

1. Останній скрипт, необхідний для забезпечення повноцінного руху Фелікса, це скрипт для стрибків. Нехай Фелікс підстрибує при натисканні клавіші пропуск.

Ось кілька порад щодо стрибків

- Фелікс не повинен падати, коли він піднімається вверх.
- Під час стрибка Фелікса не повинні зупиняти перекриття чи підлога, поки він рухається вверх, але при русі вниз він повинен зупинитись, коли торкнеться перекриття.
- Фелікс не повинен підстрибувати, вдаряючись об нижню частину зелених перешкод.
- Ми хочемо, щоб стрибки були анімовані, а не щоб Фелікс миттєво піднімався на значну висоту.

- Фелікс може підстрибнути, тільки якщо він стоїть на твердій поверхні. Гра була б занадто простою, якби Фелікс міг стрибати просто у повітрі.

1. Стрибки контролюватимуться за допомогою нової змінної - «висота стрибка». Якщо вона > 0, Фелікс рухається вгору. Якщо вона дорівнює нулю, Фелікс падає (або впав), до попереднього рівня.
2. За інших рівних умов нехай висота стрибка Фелікса дорівнює 100 пікселів. Додайте ще один цикл «якщо» всередині блоку «завжди», який реагує на натискання клавіш. Якщо в момент натискання клавіші пропуск Фелікс стоїть на перекриті (тобто змінна блокувати низ має значення 1), встановіть висоту стрибка в 100.
3. Потрібно також змінити сценарій, прописаний для падіння. Усередині блоку «завжди», створимо цикл «якщо-інакше», який виявлятиме, стрибає Фелікс вгору чи ні. Цей «якщо-інакше» цикл буде виконуватись, коли висота стрибка більша нуля. Раніше написаний скрипт для падіння тепер запишемо у частину «інакше» циклу «якщо-інакше».
4. Коли ми знаємо, що Фелікс стрибнув вгору, ми повинні перевірити, чи не вдарився він головою об щось. Якщо змінна «блокувати верх» рівна 1, встановіть висоту стрибка рівною нулю. (Це не дозволить Феліксу стрибати на перешкоди). **В іншому випадку слід перемістити Фелікса на 10 вгору на десять і зменшити висоту стрибка на десять.**
5. У результаті повинен вийти такий скрипт:

```
коли натиснуто зелений прапорець (handle falling)
завжди
 якщо <висота стрибка=0>
 якщо блокувати верх=1
 встановити висота стрибка в 0
 інакше
 зімнити у на 10
 змінити висота стрибка на -10
 кінець якщо
 інакше
 якщо <блокувати низ=0>
 змінити на -2
 кінець якщо
 кінець якщо
кінець завжди
```

Протестуйте свій проект

Натисніть зелений прапорець. Фелікс стрибає? Чи перестрибує він з однієї платформи на іншу? Чи падає він, коли досягає кінця платформи? Що відбувається, коли він стрибає з краю платформи? Що відбувається, коли він намагається стрибнути під зелений блок справа? Що відбувається при натисканн клавіші пропуск в момент падіння Фелікса?

Збережіть свій проект.

Завдання 4: Ключі та цілі

Фелікс може рухатись в межах світу. Як зробити так, щоб він завершив проходження рівня? Навколо печери знаходяться три ключі. Фелікс збирає ключі, торкаючись їх. Коли він збере всі три ключі, він зможе побачити лаз з печери і втекти.

1. Нам треба створити ще одну змінну, за допомогою якої ми відслідковуватимемо, скільки ще ключів треба зібрати. Новий скрипт для сцени повинен встановити її значення рівним 3 при натисненні зеленого прапорця.
2. Ключі та лаз з печери - це окремі спрайти. Використайте спрайти з папки для проекту несамовитий фелікс/ключ (frantic-felix/key) та несамовитий фелікс/лаз для втечі (frantic-felix/escape-pod).
3. Для кожного ключа слід прописати два скрипти: один з них відповідатиме за правильне місце розташування ключа, розмір та кут нахилу і міститиме в циклі «завжди» команду зміни кольору (щоб ключ легко було помітити на екрані). Інший скрипт з циклом «завжди» фіксуватиме момент, коли Фелікс торкнеться ключа. Як тільки це станеться, ключ повинен щезнути, а кількість ключів, які ще треба зібрати, зменшитись на один.
4. Скрипт лаза для втечі буде дещо складнішим. У ньому потрібно використати команди завжди і якщо, які чекатимуть, коли всі змінні ключів стануть рівними нулю. Як тільки це відбудеться, спрайт лаз для втечі почне блимати (щоб гравець побачив вихід). Потім можна використати інший блок якщо, який фіксуватиме момент дотику Фелікса до лаза. Коли Фелікс його торкнеться, лаз виводить на екран повідомлення «Ти виграв!». Фелікс, побачивши його, щезає.

```
коли натиснуто зелений прапорець
переміститись в x:220 y:-125
завжди якщо <ключ = 0>
 змінити ефект колір на 25
 якщо <доторкається Фелікс?>
 оповістити виграв
 говорити Ти виграв
 кінець якщо
кінець завжди якщо
```

Протестуйте свій проект

Натисніть на зелений прапорець

Збережіть свій проект.

Завдання 5: Вороги і перешкоди, що є елементами фону

Тепер про небезпечні предмети!

Є два типи небезпечних предметів. По-перше, це вороги, які рухаються навколо Фелікса і можуть вбити його, коли він на них наткнеться. По-друге, це перешкоди, які є частинами фону.

Спершу створимо персонажа-ворога. Він просто рухатиметься визначеним маршрутом.

Завдання для виконання

1. Створіть новий спрайт, обравши будь-який образ для нього. Зробіть його розмір таким само, як у Фелікса (ми використали образ з папки Речі ваза для квітів, відредагований у бік зменшення в чотири прийоми). В одному скрипті буде прописано і рух персонажа-ворога, і фіксація моменту зіткнення з Феліксом.
2. У циклі «завжди» пропишіть три блоки «якщо». Перший блок перевірятиме, чи торкнувся Фелікс ворога; якщо це так, то він виводитиме оповіщення про програш. Інші два блоки «якщо» перевірятимуть, чи пройшов ворог весь свій маршрут; якщо так, то він обертатиметься і буде йти назад. Врешті, ворог робитиме лише дві дії. (Використання команди "переміститись", а не команди "ковзати" спрощує контроль за швидкістю руху персонажа-ворога). Нам не потрібно використовувати детектор зіткнень спрайтів тут, бо нам байдуже, якою стороною Фелікс наткнеться на ворога.

```
коли натиснуто зелений прапорець
переміститисс в x:-50 y:47
повернути в напрямку -90
завжди
 якщо <доторкається Фелікс?>
 оповістити програш
 кінець якщо
 якщо x позиція > -200
 повернути в напрямку 90
 кінець якщо
 якщо x позиція > -50
 повернути в напрямку -90
 кінець якщо
 переміститись на 2 кроків
кінець завжди
```

3. Додайте скрипти до спрайтів Фелікса та лазу для втечі, щоб на екран виводилися сповіщення про програш. Фелікс повинен щезати у відповідь на отримання сповіщення, а лаз повинен виводити повідомлення "Ти програв!".

Протестуйте свій проект

Натисніть зелений прапорець. Does the vase move along? Ваза рухається? Вона зупиняється і робить поворот, коли досягає кінця свого маршруту? Що відбувається,

коли Фелікс натикається на неї? Що трапиться, коли Фелікс стрибне на неї зверху чи знизу? Чи зникає Фелікс? Чи виводить спрайт лаза для втечі правильне оповіщення? Гру можна виграти?

Збережіть свій проект.

1. **Тепер перейдемо до перешкод, що є елементами фону.** Нехай будь-що яскраво-блакитний буде небезпечним для Фелікса. Завантажте фон шалений-фелікс/рівень2, на верхньому рівні якого є блакитна троянда. Додайте ще один скрипт до спрайту Фелікса: коли натиснуто зелений прапорець завжди якщо доторкається кольору блакитний оповістити ти програв.

Протестуйте свій проект

Натисніть зелений прапорець. Фелікс помирає, коли торкається блакитної троянди? Що відбувається, коли він торкається інших речей?

Збережіть свій проект.

Короткий підсумок

Створений вами проект – це дуже проста ігрова платформа. У такому вигляді гра, звичайно, примітивна. Але не в цьому справа. Важливо, що під час її створення ви навчились писати скрипти, які допоможуть вам створити власні ігрові платформи. Наступного тижня ви розробите свої рівні ігор.

Тиждень 2: Розробка рівнів гри

Минулого тижня ви створили всі частини ігрової платформи. Цього тижня ви скористаєтесь ними для розробки рівнів власної гри.

Нагадаємо, що ви навчились писати скрипти, при виконанні яких:

- Фелікс може рухатись вліво, вправо і стрибати;
- Фелікс падає, поки його не зупинить чорне перекриття;
- Фелікс не може проходити крізь зелені перешкоди;
- блакитні елементи фону і вороги вбивають Фелікса;
- вороги рухається по одному й тому ж маршруту, як патрульні;
- вороги вбивають Фелікса, якщо він їх торкнеться;
- якщо Фелікс торкається ключа, він отримує його;
- коли Фелікс отримує всі ключі, стає видимим лаз для втечі і Фелікс переміщується у безпечне місце (або на наступний рівень).

Прогляньте ці поради. Скористайтесь ними для створення своїх власних рівнів гри.

Ви хочете створити кілька рівнів, які Фелікс повинен пройти. Наступного тижня ви побачите, як їх об'єднати. Рівні можуть мати великі або маленькі платформи, платформ може бути багато або ж декілька. Ворогів у Фелікса також може бути дуже багато або не бути зовсім. Гра може містити багато перешкод і небезпечних елементів фону.

Не обмежуйте рівень єдино можливим варіантом його успішного проходження, хай у ньому буде кілька маршрутів, які ведуть до перемоги, навіть якщо один з них простіший за інші. Подумайте про те, наскільки складним чи простим буде рівень.

Ви можете замінити спеціальні кольори (чорний, зелений і блакитний), але тоді вам необхідно буде оновити також і кольори в усіх командах "доторкається кольору" усіх скриптів. На всіх рівнях кольори, обрані вами, повинні бути одними й тими ж. (Ви можете застосовувати різні кольори на різних рівнях, але для цього доведеться прописати багато команд "або" поруч з командами "доторкається кольору".)

Протестуйте свої рівні. Якщо маєте час, створіть свої рівні у Скретч і спробуйте їх пройти. Переконайтесь, чи вони не занадто складні//прості. Якщо ви застосуєте рівні, перевірте, чи зберегли ви фон і відзначте початкову позицію Фелікса, ключів та перешкод. Переконайтесь, що ви записали, куди і на яку відстань рухаються вороги Фелікса.

Якщо ви уже розробили кілька рівнів і створили їх у Скретч, виконайте ще такі завдання:

Додатково: Наступити на ворога

Чому б не зробити так, щоб ворог Фелікса гинув, коли наступити на нього ногою? Можна додати до спрайтів ворогів скрипт, при виконанні якого з ними щось відбувається, коли вони торкаються детектора зіткнення внизу спрайта головного героя.

Додатково: Пілюлі могутності

Ви можете створити пілюлі могутності, які дозволять Феліксу знищувати ворогів. Коли Фелікс підбере пілюлю могутності, він може знищити будь-якого ворога, якого торкнеться. Дія пілюлі за деякий час зникає.

Ви можете самостійно вирішити, яким чином працюватиме пілюля. Вороги, наприклад, можуть змінювати колір на час, коли Фелікс знаходиться під дією пілюлі і може їх вбити.

Протестуйте свій проект

Натисніть зелений прапорець.

Збережіть свій проект.

Тиждень 3: Створення цілісної гри.

У вас є кілька рівнів для гри. Ви знаєте достатньо команд для того, щоб усі ці рівні працювали. Залишилось тільки поєднати їх в одну гру. Якщо ви можете швидко поєднати рівні гри, прогляньте завдання 2 і 3 до того, як почнете грати в гру.

Завдання 1: Відображення нового рівня

Коли Фелікс пройде один рівень, він повинен перейти на інший. Це означає, що у момент знайдення лазу для втечі спрайт лазу повинен вивести повідомлення про перехід на інший рівень (а не про перемогу). Повідомлення про перехід на інший рівень може слугувати для запуску наступного рівня. Потрібна також змінна "поточний рівень" для спрайту **Лаз**, яку він виводитиме до повідомлення, що запускатиме наступний рівень.

Майже всі складові повинні зреагувати на повідомлення про **перехід на інший рівень**.

На сцені повинно з'явитись відповідне тло. Усі вороги, ключі, лаз для втечі повинні перемістись у правильні позиції. Маршрути руху ворогів повинні оновитись. Фелікс повинен перемістись у свою нову стартову позицію. Тоді гра на новому рівні може розпочинатись.

Більшість спрайтів повинні будуть зреагувати на повідомленні про **перехід на новий рівень**, а не на натиснення зеленого прапорця. Тобто для більшості скриптів треба буде змінити першу команду (подія, яка запускає виконання скриптів).

Збережіть свій проект

Перш ніж щось зробити, збережіть свій проект. Ви постійно вноситимете зміни у гру Шалений Фелікс, тому не забувайте час від часу зберігати її.

Протестуйте свій проект

Тестуйте гру щоразу після внесення змін. Тестуйте ті частини, у які внесено зміни, щоб переконатись, що вони працюють правильно. Ми не будемо розповідати, як вносити усі зміни. Однак ми покажемо, як виглядає оновлений скрипт для Фелікса, щоб ви зрозуміли, що саме треба буде зробити.

```
коли я отримаю [почати рівень]
переміститись в x:(елемент поточний рівень з [xs])y:( елемент поточний рівень з [ys])
повернути в напрямку (елемент `поточний рівень` з [напрямки])
показати
```

```
завжди
 якщо << клавішу [стрілка ліворуч] натиснуто?> і `блокувати ліворуч`=[0]>
 повернути в напрямку -90
 переміститись на 2 кроків
 кінець якщо
 якщо << клавішу [стрілка праворуч] натиснуто?>і `блокувати праворуч`=[0]>
 повернути в напрямку 90
 переміститись на 2 кроків
 кінець якщо
 якщо <<клавішу[пропуск] натиснуто?>і `блокувати низ`=[1]>
 встановити [висоат стрибка]в[100]
 кінець якщо
кінець завжди
```

Зверніть увагу, що початкові значення **x**, **y**, і напрям для Фелікса задаються списками. Для кожного спрайту створено кілька списків (кожен список тільки для певного спрайту), щоб зберігати значення, які потрібні будуть для цього спрайта. Для кожного значення потрібен один список. Замість списків можна використати команду "якщо", яка перевірятиме поточний рівень і виконуватиме правильну дію з огляду на його значення.

Ось і скрипт лаза для втечі, який відповідає за перехід на нові рівні:

```
коли натиснуто зелений прапорець
встановити [поточний рівень] в 1
оповістити [почати рівень]

коли я отримаю [почати рівень]
переміститись в x:(елемент поточний рівень з [xs])y:( елемент поточний рівень з [ys])
завжди
 якщо <якщо ключ = 0>
 змінити ефект колір на [25]
 якщо <доторкається [Фелікс]?>
 якщо <поточний рівень = `довжина [ключів на рівні]>
 говорити [Ти виграв!]
 оповістити [виграш]
 зупинити все
 інакше
 змінити [поточний рівень] на 1
 оповістити [почати рівень]
 кінець якщо
 кінець якщо
 кінець якщо
кінець завжди
```

Завдання 2: Грай!

Ти створив гру. Твої друзі по Code Club теж створили свої ігри. Грайте в них! Тобі вдається виграти у чужих іграх? Чи зміг хтось перемогти у твоїй грі?

Додаково: Кілька життів

У Фелікса повинно бути кілька спроб або кілька життів для проходження усіх рівнів. Створіть новий спрайт з трьома образами, які показують одне, два і три серця. Використайте образи шалений-фелікс/1-серце, шалений-фелікс/2-серце і шалений-фелікс/3-серце. Цей спрайт повинен з'явитися в кутку сцени. При натисканні зеленого прапорця повинно бути видно три серця. Кожного разу після отримання повідомлення про програш, повинно ставати на одне серце менше. Коли кількість сердець стане 0, спрайт повинен щезнути, залишивши повідомлення про завершення гри.

Повідомлення про завершення гри та перемогу повинен показувати новий спрайт, який щезає при натисканні зеленого прапорця і показує правильний напис при отриманні оповіщень про завершення гри і перемогу в грі. Цей спрайт також повинен зупиняти виконання всіх скриптів на час, коли він стає видимим.

Також треба відредагувати скрипт спрайта **Лаз для втечі** (який раніше містив надсилання повідомлень про виграш та програш), оскільки це тепер буде прописано у скриптах спрайтів життя та завершення гри.

Може статись так, що персонажі ворогів занадто швидко з'являються перед Феліксом (ще до того, як новий рівень повністю запуститься). Якщо Фелікс втрачає більше, ніж одне життя, коли натикається на ворога, сховайте ворога, як тільки він надішле повідомлення про програш. Це дасть можливість грі (в тому числі Феліксу) повністю перезапуститись до того, як ворог виявить чергове зіткнення з Феліксом.

Додатково: Обмеження часу

Обмежте час гри для Фелікса. За допомогою **скретч картки Таймер** введіть обмеження часу. Коли час буде вичерпано, пропишіть появу повідомлення про програш. Не забувайте перезапускати таймер при переході на новий рівень.