МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «УНІВЕРСИТЕТ «КРОК»
Фаховий коледж Університету «КРОК»

ДИПЛОМНА РОБОТА
за темою
«Розробка та створення відеогри»

Студент 4 курсу групи ІПЗ-20к/2 **Керівник дипломної роботи**

(посада керівника)

Чабала Єгор Костянтинович — Пантєєв Роман Леонідович

(прізвище, ім'я та по-батькові студента) (прізвище, ім'я та по-батькові керівника)

До захисту

(резолюція «До захисту»)

10.06.2024

(підпис студента) **(дата)** **(підпис викладача)**

Київ, 2024 рік

ЗМІСТ

РОЗДІЛ 1. ТЕОРЕТИЧНА ЧАСТИНА РОЗРОБКИ ТЕКСТОВИХ ВІДЕОІГОР В ЖАНРІ METROIDVANIA ЗА ДОПОМОГОЮ PYTHON, PYGAME

1.1 Огляд історії текстових відеоігор та жанру Metroidvania

Text-based ігри:

Історія text-based ігор бере свій початок ще з ранніх днів комп'ютерних технологій. Ось основні етапи розвитку цього жанру:

1960-ті роки: Перші експерименти

1961: Поява першої відомої text-based гри "Spacewar!". Хоча це була графічна гра, вона надихнула на подальший розвиток інтерактивних комп'ютерних розваг.

1966: Джозеф Вайценбаум створює програму ELIZA, яка імітувала поведінку психотерапевта. Це була одна з перших програм, що використовували текстовий інтерфейс для взаємодії з користувачем.

1970-ті роки: Народження жанру

1971: "The Oregon Trail" – одна з перших text-based ігор, розроблена для навчальних цілей. Гравці керували групою переселенців, які подорожували з Міссурі до Орегону.

1975: "Adventure" або "Colossal Cave Adventure" створена Віллом Кроутером і Дональдом Вудсом стала першою широко відомою text-based грою. Гравці досліджували підземелля, шукаючи скарби.

1977: "Zork" була створена студентами Массачусетського технологічного інституту (MIT) і стала однією з найвідоміших text-based ігор. Вона мала значний вплив на подальший розвиток жанру.

1980-ті роки: Золотий вік text-based ігор

1980: "Rogue" стала популярною text-based грою з процедурно генерованими підземеллями, яка заклала основу для піджанру "roguelike".

1981: "Zork" випускається компанією Infocom як комерційна гра і стає надзвичайно успішною.

Протягом цього десятиліття Infocom випустила багато популярних text-based ігор, включаючи "The Hitchhiker's Guide to the Galaxy" та "Planetfall".

1990-ті роки: Зміни та нові горизонти

У міру розвитку графічних технологій text-based ігри почали втрачати популярність.

Проте жанр продовжував існувати у вигляді MUD (Multi-User Dungeon), багатокористувацьких text-based ігор, які дозволяли гравцям взаємодіяти в реальному часі.

2000-ті роки та далі: Відродження інтересу

З розвитком інтернету та інді-ігор жанр text-based ігор пережив новий сплеск інтересу.

2005: "Fallen London" – популярна text-based гра, що дозволила гравцям досліджувати альтернативну версію вікторіанського Лондона.

2010-ті роки: Інді-розробники випускають безліч text-based ігор, які отримують позитивні відгуки та прихильність гравців. Наприклад, "80 Days" від Inkle, яка поєднує текстову гру з графічними елементами.

Сьогодення: Текстові ігри в епоху інді-розробок

Завдяки платформам, таким як Twine, багато розробників-аматорів створюють власні text-based ігри, часто з експериментальними сюжетами та механіками.

Гравці цінують text-based ігри за їхню здатність розповідати глибокі, інтерактивні історії, що дозволяють користувачам повністю зануритися у світ гри.

Тож, хоча графічні ігри зайняли домінуюче положення в індустрії, text-based ігри продовжують існувати та розвиватися, привертаючи увагу гравців своєю унікальною здатністю створювати інтерактивні історії та надавати можливість глибокого занурення в сюжет.

Важливі моменти в розвитку text-based ігор

Twine: Інструмент для створення інтерактивних текстових історій, який став популярним серед інді-розробників завдяки своїй простоті та гнучкості. Twine дозволяє створювати нелінійні історії та додавати елементи інтерактивності.

Interactive Fiction (IF) Community: Спільнота розробників і шанувальників інтерактивної літератури продовжує активно розвиватися. Щорічні конкурси та фестивалі, такі як IFComp, допомагають виявити нові таланти та популяризувати жанр.

Сучасні text-based ігри: Ігри, такі як "80 Days" і "Lifeline", демонструють, як text-based механіки можуть бути поєднані з сучасними технологіями та дизайном, щоб створювати захоплюючі ігрові враження. "80 Days" використовує текст для розповіді історії подорожі навколо світу, тоді як "Lifeline" пропонує інтерактивний наратив, де гравець спілкується з персонажем, що застряг на іншій планеті.

Зростання мобільних платформ: Мобільні пристрої стали ідеальною платформою для text-based ігор, надаючи гравцям можливість взаємодіяти з іграми під час поїздок або у вільний час. Багато text-based ігор доступні на смартфонах та планшетах, що робить їх доступнішими для широкої аудиторії.

Зростання інтересу до ретро-ігор: Відродження інтересу до ретро-ігор також сприяло поверненню text-based ігор. Гравці ностальгічно згадують простіші часи, коли текстові ігри були на піку популярності, і охоче занурюються в нові проекти, натхненні класикою.

Отже, історія text-based ігор багата і різноманітна. Вони пережили кілька етапів розвитку, від перших експериментів у 1960-х роках до сучасного відродження інтересу завдяки інді-розробникам та новим технологіям. Незважаючи на зміни у вподобаннях гравців та розвиток графічних технологій, text-based ігри продовжують залишатися важливою частиною ігрової індустрії, пропонуючи унікальні способи взаємодії з інтерактивними історіями.

Metroidvania ігри:

Історія ігор в жанрі Metroidvania є цікавою та багатогранною. Жанр отримав свою назву від двох культових серій відеоігор: "Metroid" від Nintendo та "Castlevania" від Konami. Давайте розглянемо історію цього жанру більш детально.

Початок: Виникнення жанру

Metroid

Першою грою, яка заклала основи жанру, стала "Metroid", випущена компанією Nintendo у 1986 році для консолі Famicom Disk System, а згодом для Nintendo Entertainment System (NES). Гравці керували Самус Аран, космічним мисливцем, досліджуючи інопланетні світи, змагаючись з ворогами та збираючи різні покращення та зброю, що дозволяли доступ до нових областей. Характерною особливістю гри стала нелінійність та потреба постійно повертатися до вже пройдених локацій, щоб відкрити нові зони за допомогою щойно знайдених предметів або здібностей.

Castlevania

Через кілька років після виходу Metroid, Konami випустила "Castlevania" (1986), яка стала ще однією значущою грою в історії жанру. Однак, саме "Castlevania: Symphony of the Night" (1997) для PlayStation закріпила жанр Metroidvania, додавши елементи RPG (рівні, очки досвіду та інвентар), що зробило гру ще глибшою та багатшою на контент.

Еволюція жанру

1990-ті роки

Протягом 1990-х років багато ігор приймали концепцію Metroidvania, але саме "Castlevania: Symphony of the Night" стала вирішальною у популяризації жанру. Ця гра запровадила багатошаровий підхід до дизайну рівнів, велику кількість секретів та розширений набір можливостей для гравця, що значно вплинуло на майбутні проекти.

2000-ті роки

На початку 2000-х жанр почав розширюватися. Вийшли такі значущі ігри, як "Metroid Fusion" (2002) та "Metroid Zero Mission" (2004) для Game Boy Advance. Серія "Castlevania" продовжувала виходити на різних платформах, підтримуючи та розвиваючи жанр.

Незалежні розробники

Великий вплив на розвиток жанру мали незалежні розробники. Вони взяли базові принципи Metroidvania та додали власні ідеї, експериментуючи з механіками та стилями. Прикладами таких ігор є:

"Cave Story" (2004): Незалежна гра, розроблена Daisuke "Pixel" Amaya. Вона поєднала в собі елементи платформера та RPG, і стала культовою серед гравців.

"Shadow Complex" (2009): Випущена Epic Games, ця гра стала популярною на Xbox Live Arcade, привернувши увагу до жанру з боку нової аудиторії.

"Hollow Knight" (2017): Гра від незалежної студії Team Cherry, яка здобула значну популярність завдяки своєму глибокому світу, складним боям та високій якості виконання.

Сучасний стан жанру

Жанр Metroidvania продовжує розвиватися та процвітати. Випускаються нові ігри, які впроваджують сучасні технології та ідеї:

"Ori and the Blind Forest" (2015) та "Ori and the Will of the Wisps" (2020): Від студії Moon Studios, ці ігри здобули похвалу за свою чудову графіку, музику та ігровий процес.

"Axiom Verge" (2015): Випущена незалежним розробником Томом Хаппом, ця гра отримала високу оцінку за свою вірність класичним принципам жанру Metroidvania та інноваційний підхід до геймплею.

"Bloodstained: Ritual of the Night" (2019): Від Кої Ігарасі, одного з творців "Castlevania: Symphony of the Night", ця гра продовжила традиції жанру, додавши нові елементи та сучасну графіку.

Жанр Metroidvania еволюціонував від ранніх експериментів у нелінійному дизайні рівнів до складних, глибоких та захоплюючих світів, які ми бачимо сьогодні. Він постійно змінюється та адаптується, залучаючи нових гравців та надихаючи нові покоління розробників. Незалежні ігри та великі проекти продовжують збагачувати жанр, роблячи його одним з найцікавіших та найрізноманітніших у світі відеоігор.

1.2 Використання мови програмування Python для розробки відеоігор

Мова програмування Python стала популярною у багатьох галузях, включаючи розробку відеоігор. Її простота у використанні, велика кількість бібліотек та активна спільнота роблять Python привабливим вибором для розробників. Розглянемо детальніше, як Python використовується у розробці відеоігор.

Основи використання Python для розробки відеоігор

1. Python

Python - це інтерпретована мова програмування високого рівня, відома своєю читабельністю та простотою синтаксису. Вона підтримує парадигми об'єктно-орієнтованого, процедурного та функціонального програмування, що робить її універсальною для різних типів проектів, включаючи розробку ігор.

2. Переваги використання Python для розробки ігор

1. Легкість вивчення та використання**: Простий синтаксис Python дозволяє швидко почати розробку ігор навіть новачкам.

2. Багатий набір бібліотек та фреймворків**: Python має безліч бібліотек та фреймворків, що спрощують розробку ігор.

3. Активна спільнота**: Велика спільнота розробників, яка допомагає вирішувати проблеми та ділиться своїми напрацюваннями.

4. Кросплатформеність**: Python працює на різних операційних системах, що дозволяє розробляти гри для різних платформ.

Основні бібліотеки та фреймворки для розробки ігор на Python

1. Pygame

Pygame - це одна з найпопулярніших бібліотек для розробки 2D-ігор на Python. Вона побудована на основі бібліотеки SDL (Simple DirectMedia Layer) і надає розробникам інструменти для роботи з графікою, звуком та введенням користувача.

- Основні можливості:
  - Робота з графікою (спрайти, анімації, текстури).
  - Обробка введення з клавіатури, миші та джойстика.
  - Відтворення звукових ефектів та музики.
  - Створення ігрових циклів та управління часом.

- Приклад коду:

```
import pygame

pygame.init()
screen = pygame.display.set_mode((800, 600))
pygame.display.set_caption('Simple Pygame Example')

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
```

```
        screen.fill((0, 0, 0))
        pygame.display.flip()

    pygame.quit()
```

2. Pyglet

Pyglet - це бібліотека для розробки ігор та мультимедійних додатків, яка дозволяє працювати з OpenGL. Вона підходить для створення більш складних 2D- та 3D-ігор.

Основні можливості:

Підтримка OpenGL для роботи з 3D-графікою.

Обробка текстур, шейдерів та спрайтів.

Підтримка різних форматів аудіо та відео.

Обробка введення з клавіатури, миші та джойстика.

Panda3D - це потужний движок для розробки 3D-ігор, який підтримує Python та C++. Він використовується для створення як простих, так і складних тривимірних проектів.

Основні можливості:

Підтримка повного набору інструментів для роботи з 3D-графікою.

Вбудовані системи фізики та анімації.

Можливість створення як одиночних, так і багатокористувацьких ігор.

Підтримка VR та AR технологій.

4. Godot (з підтримкою GDScript на основі Python)

Godot - це популярний відкритий движок для розробки 2D- та 3D-ігор. Основною мовою для скриптів у Godot є GDScript, але існує можливість використовувати і Python за допомогою модуля Godot-Python.

Основні можливості:

Зручний інтерфейс для розробки ігор.

Підтримка анімації, фізики та системи часток.

Підтримка різних платформ: Windows, macOS, Linux, Android, iOS та інші.

Можливість розробки VR- та AR-додатків.

1. "Frets on Fire": Музична гра, створена з використанням Pygame. Гравці грають на гітарі, натискаючи клавіші у ритм музиці.

2. "PySol": Велика колекція пасьянсів, написана на Python, яка використовує Tkinter для графічного інтерфейсу.

3. "Disney's Toontown Online": Багатокористувацька онлайн-гра, розроблена з використанням Panda3D.

Python пропонує багатий набір інструментів та бібліотек для розробки відеоігор, що робить його популярним вибором серед розробників. Незалежно від того, чи плануєте ви створювати прості 2D-ігри або складні 3D-проекти, Python надає всі необхідні засоби для досягнення ваших цілей. Завдяки своїй простоті, кросплатформенності та активній спільноті, Python залишається актуальним вибором для розробки відеоігор.

1.3 **Бібліотека Pygame: основні можливості та функціонал**

Pygame — це популярна бібліотека для розробки 2D-ігор на Python. Вона побудована на основі бібліотеки SDL (Simple DirectMedia Layer) і надає розробникам широкий набір інструментів для роботи з графікою, звуком, введенням користувача та іншими аспектами ігрового процесу. Нижче представлено детальний огляд можливостей та функціоналу Pygame.

Основні можливості Pygame

1. Робота з графікою

Pygame надає засоби для створення і управління графічними об'єктами, такими як спрайти, текстури, анімації та інші візуальні елементи.

Спрайти: Pygame має потужну систему спрайтів, що дозволяє легко створювати і управляти рухомими об'єктами на екрані.

Поверхні (Surfaces): Поверхні в Pygame використовуються для відображення зображень і текстур. Ви можете створювати нові поверхні, завантажувати зображення з файлів і малювати на поверхнях.

Анімації: Можливість створення анімацій за допомогою зміни спрайтів або поверхонь у часових інтервалах.

Обробка зображень: Включає масштабування, обертання, відображення та інші трансформації зображень.

2. Обробка введення

Pygame дозволяє обробляти введення з клавіатури, миші, джойстиків та інших пристроїв.

Клавіатура: Обробка подій натискання клавіш, відстеження натиснутих клавіш та управління текстовим вводом.

Миша: Відстеження позиції миші, обробка натискань кнопок миші та подій пересування.

Джойстики та геймпади: Підтримка підключення та обробки подій від різних контролерів.

3. Звук та музика

Pygame надає можливості для роботи зі звуками та музикою, дозволяючи створювати багатий аудіо супровід для ігор.

Відтворення звукових ефектів: Завантаження та відтворення коротких звукових ефектів.

Фонова музика: Завантаження та відтворення музичних треків на фоні.

Звукові канали: Управління звуковими каналами для відтворення одночасно кількох звукових ефектів.

4. Таймери та події

Pygame має систему таймерів та подій, що дозволяє управляти ігровими циклами, створювати часові затримки та обробляти користувацькі події.

Ігровий цикл: Виконання основних дій гри у циклі, що постійно оновлюється.

Таймери: Створення таймерів для відстеження часу або створення повторюваних подій.

Події: Обробка подій введення, таймерів та користувацьких подій.

Приклад коду

Простий приклад гри на Pygame:

```
import pygame
import sys

# Ініціалізація Pygame
pygame.init()

# Налаштування вікна
screen = pygame.display.set_mode((800, 600))
pygame.display.set_caption('Simple Pygame Example')

# Колір
black = (0, 0, 0)
white = (255, 255, 255)
```

```
# Основний ігровий цикл
running = True
while running:
    # Обробка подій
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Очищення екрану
    screen.fill(black)

    # Малювання білого кола
    pygame.draw.circle(screen, white, (400, 300), 50)

    # Оновлення екрану
    pygame.display.flip()

# Завершення Pygame
pygame.quit()
sys.exit()
```

Додаткові функціональні можливості

1. Робота з текстом

Pygame дозволяє рендерити текст на поверхнях за допомогою вбудованого модуля `pygame.font`.

Створення шрифтів: Можливість використовувати стандартні шрифти або завантажувати власні.

Рендеринг тексту: Відображення тексту на поверхнях з налаштуванням кольору, розміру та інших параметрів.

2. Обробка зіткнень

Pygame має вбудовані функції для обробки зіткнень між спрайтами та іншими об'єктами.

Прямокутні зіткнення: Використання прямокутних областей для визначення зіткнень.

Маски зіткнень: Точніше визначення зіткнень за допомогою масок, що враховують форму об'єктів.

3. Мережевий режим

Pygame дозволяє створювати мережеві ігри, використовуючи стандартні засоби для роботи з мережами на Python.

Клієнт-серверна архітектура: Можливість створення як клієнтських, так і серверних частин гри.

Передача даних: Відправка та отримання даних між гравцями у мережевому середовищі.

4. Вбудовані модулі

Pygame включає в себе низку вбудованих модулів, що спрощують розробку ігор:

pygame.draw: Набір функцій для малювання простих геометричних фігур (ліній, кола, прямокутників тощо).

pygame.image: Завантаження та збереження зображень у різних форматах.

pygame.mixer: Управління звуковими ефектами та музикою.

pygame.time: Управління часом та таймерами.

pygame.sprite: Клас для роботи зі спрайтами та групами спрайтів.

pygame.locals: Збереження часто використовуваних констант для зручності.

Переваги та недоліки Pygame

Переваги

1. Легкість вивчення: Простий синтаксис Python та добре задокументована бібліотека Pygame дозволяють швидко почати розробку ігор.

2. Широкий функціонал: Підтримка графіки, звуку, введення та інших аспектів ігрового процесу.

3. Кросплатформеність: Pygame працює на різних операційних системах, включаючи Windows, macOS та Linux.

4. Активна спільнота: Велика спільнота розробників, що створює додаткові ресурси, посібники та приклади.

Недоліки

1. Обмежена продуктивність: Pygame більше підходить для створення 2D-ігор та простих 3D-проектів. Для складніших 3D-ігор можуть знадобитися інші движки.

2. Відсутність вбудованого редактора: Pygame не має власного редактора ігрових сцен або рівнів, що може ускладнити розробку великих проектів.

Pygame є потужним інструментом для розробки 2D-ігор на Python. Він пропонує широкий набір можливостей для роботи з графікою, звуком, введенням користувача та іншими аспектами ігрового процесу. Завдяки простоті використання та активній спільноті, Pygame є відмінним вибором як для початківців, так і для досвідчених розробників, які бажають швидко створювати ігрові проекти.

1.4 **Переваги розробки текстових відеоігор у жанрі Metroidvania**

Розробка текстових відеоігор у жанрі Metroidvania має низку переваг та є унікальним поєднанням, яке може запропонувати інноваційний підхід до створення ігор. Ось детальний розгляд переваг такого підходу:

Переваги розробки текстових відеоігор у жанрі Metroidvania

1. Низькі витрати на розробку

Текстові ігри зазвичай мають менші витрати на розробку порівняно з графічними іграми. Відсутність необхідності створення складних графічних активів, анімацій та звукових ефектів дозволяє зосередитися на створенні якісного контенту та ігрової механіки.

2. Зосередження на сюжеті та атмосфері

Текстові ігри дозволяють розробникам зосередитися на написанні глибокого та цікавого сюжету. Жанр Metroidvania, відомий своєю нелінійністю та дослідженням, може отримати новий вимір завдяки текстовій подачі, де гравці використовують уяву для створення візуальних образів ігрового світу.

3. Інтерактивність та нелінійність

Текстові ігри надають широкий простір для інтерактивності та нелінійного розвитку подій. Гравці можуть приймати рішення, що впливають на подальший розвиток сюжету, досліджувати різні шляхи та взаємодіяти з різними персонажами та об'єктами, що робить гру більш захоплюючою та непередбачуваною.

4. Легкість у створенні оновлень та розширень

Оскільки текстові ігри не потребують складної графіки, розробники можуть легко додавати нові рівні, персонажів, сюжетні лінії та інші елементи, що дозволяє підтримувати інтерес гравців і постійно оновлювати контент.

5.Інклюзивність

Текстові ігри доступні для більш широкої аудиторії, включаючи людей з обмеженими можливостями зору. За допомогою екранних читалок та інших асистивних технологій ці ігри можуть бути адаптовані для гравців з різними потребами.

6. Можливість реалізації унікальних механік

Поєднання жанру Metroidvania з текстовим форматом дозволяє експериментувати з унікальними ігровими механіками, які не завжди можливо реалізувати в графічних іграх. Наприклад, гравці можуть використовувати текстові команди для переміщення, дослідження та взаємодії з оточенням.

Інноваційність поєднання

1. Збагачення жанру Metroidvania

Традиційно ігри Metroidvania орієнтовані на дослідження та прогрес через отримання нових здібностей. Текстовий формат додає новий шар глибини, дозволяючи гравцям використовувати власну уяву та інтерпретацію, що може зробити ігровий досвід більш особистим та захоплюючим.

2. Нові способи розкриття історії

Текстовий формат дозволяє детальніше розкривати історію, персонажів та світ гри через описові тексти, діалоги та внутрішні монологи. Це дозволяє

створювати багатошарові та складні наративи, які можуть мати великий вплив на гравців.

3. Гнучкість у дизайні гри

Текстові ігри надають розробникам гнучкість у дизайні рівнів та механік. Відсутність графічних обмежень дозволяє створювати складні та варіативні ігрові світи, які можна легко адаптувати та розширювати.

4. Експериментальні наративні структури

Поєднання жанру Metroidvania з текстовим форматом відкриває можливості для експериментальних наративних структур, де гравці можуть досліджувати не лише фізичні простори, а й складні історії та концепції через текстові взаємодії.

5. Використання природної мови

Використання тексту дозволяє інтегрувати елементи природної мови, що робить взаємодію з грою більш природною та інтуїтивною. Гравці можуть використовувати команди на природній мові для взаємодії з грою, що робить процес більш захоплюючим та інтерактивним.

Розробка текстових відеоігор у жанрі Metroidvania має численні переваги та відкриває нові горизонти для інновацій у створенні ігор. Поєднання глибокого сюжету, інтерактивності, нелінійного розвитку та унікальних механік робить ці ігри особливими та захоплюючими для широкого кола гравців. Це унікальне поєднання дозволяє розробникам створювати незабутні ігрові враження, зосереджуючись на історії та атмосфері, що робить текстові Metroidvania-ігри інноваційним та цікавим напрямом у розвитку індустрії відеоігор.

РОЗДІЛ 2. ОГЛЯД ІСНУЮЧИХ ТЕКСТОВИХ ВІДЕОІГОР ТА РЕАЛІЗАЦІЯ В ЖАНРІ METROIDVANIA

2.1 Аналіз основоположників текстових відеоігор Переваги розробки текстових відеоігор у жанрі Metroidvania

Аналіз основоположників текстових відеоігор надає розуміння того, як цей жанр еволюціонував і вплинув на сучасну ігрову індустрію. Перші текстові відеоігри, створені ще у 1970-х роках, заклали фундамент для багатьох сучасних ігрових механік і наративних структур. Розглянемо детальніше ключові проекти і їхні впливи.

Перші текстові відеоігри

Adventure (1976)

Adventure (також відома як Colossal Cave Adventure) є однією з перших текстових відеоігор, розроблених Віллом Кроутером і пізніше розширеною Доном Вудсом. Гра пропонувала гравцям дослідити печеру, використовуючи текстові команди для взаємодії з середовищем.

Основні риси:

Гравці використовували команди на кшталт "go north" або "take lantern".

Впровадження базових елементів адвенчурних ігор, таких як вирішення головоломок і збирання предметів.

Значний вплив на розвиток жанру інтерактивної художньої літератури.

Zork (1980)

Zork — це серія текстових пригодницьких ігор, розроблених компанією Infocom. Перша гра, Zork I, стала дуже популярною і є прикладом вдалої комерційної текстової гри.

Основні риси:

Більш розвинена система команд, дозволяючи гравцям використовувати складні фрази.

Розгалужений сюжет і інтерактивні головоломки.

Значний вплив на текстові та графічні адвенчури, а також на інтерактивну літературу.

Вплив та інновації

Інтерактивність та вибір

Ранні текстові ігри зосереджувались на інтерактивності та наданні гравцям можливості впливати на сюжет. Це стало основою для багатьох сучасних ігор, де вибір гравця є центральним елементом.

Вплив на сучасні ігри: Багато сучасних ігор, такі як The Witcher 3 або Mass Effect, використовують вибір гравця як основний механізм наративу.

Головоломки та дослідження

Ранні текстові ігри, такі як Adventure та Zork, зосереджувались на вирішенні головоломок і дослідженні ігрового світу. Ці елементи стали основними в багатьох жанрах, включаючи пригодницькі ігри та RPG.

2.2 Аналіз основоположників metroidvania відеоігор

Ранні текстові ігри, такі як Adventure та Zork, зосереджувались на вирішенні головоломок і дослідженні ігрового світу. Ці елементи стали основними в багатьох жанрах, включаючи пригодницькі ігри та RPG.

Походження назви та основні ігри-прародичі:

Термін "Metroidvania" сформувався від об'єднання назв двох впливових ігор: "Metroid" і "Castlevania".

"Metroid", розроблений Nintendo, вийшов у 1986 році для консолі NES. Ця гра визначила багато основних елементів жанру, таких як неелінійність, дослідження, збирання предметів та прогресія навичок.

"Castlevania" - серія ігор, що розпочалася в 1986 році, також мала великий вплив на жанр. "Symphony of the Night" (1997), зокрема, вважається класичною metroidvania грою і впровадила такі елементи, як RPG-механіки, великий відкритий світ та систему прогресу героя.

Основні характеристики жанру:

Неелінійність та дослідження: Гравцям дозволяється вільно досліджувати великі, заплутані світи, не обов'язково слідуючи строго визначеному шляху.

Прогресія через навички та предмети: Герой зазвичай отримує нові навички та предмети, які відкривають нові області або дозволяють подолати перешкоди.

Платформерні елементи: Часті перешкоди, які гравцеві потрібно перескакувати або обходити.

Великий, зв'язаний світ: Світ гри зазвичай є великим та деталізованим, з численними складними зв'язками між різними областями.

Бойова система та боси: Зазвичай включає в себе бої з ворогами та босами, які вимагають від гравця стратегії та вправності.

Вплив на інші жанри та сучасні ігри:

Metroidvania вплинули на багато інших жанрів, таких як інді-ігри, RPG та адвенчури.

Багато ігор, таких як "Hollow Knight", "Dead Cells", "Axiom Verge" і "Ori and the Blind Forest", беруть свої відомості з metroidvania і розвивають їх, додаючи нові ідеї та механіки.

Популярність та успішність:

Metroidvania володіють стабільною популярністю серед гравців завдяки своїй атмосфері, геймплею та можливості досліджувати великі світи.

Багато metroidvania отримали високі оцінки критиків та масове визнання, що підтверджує їх успішність серед гравців.

Аналізуючи ці аспекти, можна зрозуміти, як metroidvania стали важливим та впливовим жанром у світі відеоігор.

2.3Аналіз поєднання жанрів metroidvania та текстових відеоігор

Аналізуючи ці аспекти, можна зрозуміти, як metroidvania стали важливим та впливовим жанром у світі відеоігор.

Поєднання жанрів metroidvania і текстових відеоігор може створити цікавий і унікальний гібрид, який поєднує в собі ключові елементи обох жанрів. Давайте розглянемо детальний аналіз цього поєднання:

Неелінійність та дослідження:

Текстові відеоігри, такі як інтерактивні книги або ігри жанру "interactive fiction", можуть створювати атмосферу неелінійності, де гравці можуть обирати різні напрямки розвитку сюжету.

Ця неелінійність може бути посиленою використанням механік дослідження, характерних для metroidvania, де гравець вільно переміщується по світу та досліджує його.

Прогресія через навички та предмети:

В текстових відеоіграх прогрес може досягатися шляхом вибору різних дій та отримання різних навичок або предметів, що впливає на розвиток сюжету.

Поєднання цих механік з metroidvania може створити цікавий гібрид, де гравець отримує нові можливості та здібності, щоб досліджувати нові області текстового світу.

Великий, зв'язаний світ:

Metroidvania відомі своїми великими світами з численними переходами та зв'язками між областями.

В текстових відеоіграх ця ідея може бути реалізована через створення великого текстового світу з численними рішеннями та напрямками, які може обрати гравець.

Бойова система та боси:

Текстові відеоігри можуть використовувати бойові механіки через текстові описи боїв або вибір дій гравця в бойових сценах.

Поєднання цих механік з metroidvania може дозволити створити унікальні бої з босами, де гравець повинен використовувати свої навички та здібності для перемоги.

Популярність інтерактивних історій:

Останнім часом інтерактивні історії, такі як "80 Days" або "Choice of Robots", стають дуже популярними серед гравців.

Поєднання цих ігор з жанром metroidvania може зацікавити гравців, які шукають не тільки захопливий сюжет, але й можливість досліджувати великі світи та вирішувати різноманітні завдання.

Отже, поєднання жанрів metroidvania та текстових відеоігор може створити цікавий гібрид, який поєднує в собі найкращі аспекти обох жанрів і створює унікальний геймплейний досвід.

РОЗДІЛ 3. ПРАКТИЧНА ЧАСТИНА РОЗРОБКИ ВІДЕОГРИ В ЖАНРІ METROIDVANIA ЗА ДОПОМОГОЮ PYTHON

3.1 Код самої гри

```
import pygame
import sys
import random

pygame.init()

WIDTH, HEIGHT = 800, 800
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Text-based Metroidvania")

WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
FONT = pygame.font.Font(None, 36)

class Room:
    def __init__(self, description, items, enemies):
        self.description = description
        self.items = items
        self.enemies = enemies

class Player:
    def __init__(self, current_room):
        self.current_room = current_room
        self.inventory = []
        self.health = 200
        if self.health <= 0:
            pygame.quit()
            sys.exit()
        self.explored_rooms = {current_room.description}

    def move_to(self, new_room):
        self.current_room = new_room
```

```
        self.explored_rooms.add(new_room.description)

    def pick_up(self, item):
        if item in self.current_room.items:
            self.current_room.items.remove(item)
            self.inventory.append(item)
            if item in item_healing_values:
                self.health += item_healing_values[item]
                if self.health > 200:  # assuming 200 is the max health
                    self.health = 200

    def fight_enemy(self, enemy_id):
        for enemy in self.current_room.enemies:
            if enemy.id == enemy_id:
                enemy_health = enemy.stats["health"]
                while enemy_health > 0 and self.health > 0:
                    enemy_damage = random.randint(5, 15)
                    self.health -= enemy_damage
                    player_damage = random.randint(10, 20)
                    enemy_health -= player_damage
                    if enemy_health <= 0:
                        self.current_room.enemies = [e for e in self.current_room.enemies if
e.id != enemy_id]
                        return "Enemy defeated!"
                    if self.health <= 0:
                        return "You were defeated!"
                    if self.health <= 0:
                        pygame.quit()
                        sys.exit()
        return "Enemy not found!"

items_list = ["torch", "key", "ancient coin", "book of secrets"]

item_healing_values = {
    "torch": 10,
    "key": 5,
    "ancient coin": 20,
    "book of secrets": 30
}

enemy_types = {
```

```
    "Goblin": {"health": random.randint(50, 80), "attack": random.randint(10, 15)},
    "Skeleton": {"health": random.randint(70, 100), "attack": random.randint(8, 12)},
    "Zombie": {"health": random.randint(80, 120), "attack": random.randint(12, 18)}
}

class Enemy:
    id_counter = 0

    def __init__(self, enemy_type):
        self.type = enemy_type
        self.stats = enemy_types[enemy_type]
        self.id = Enemy.id_counter
        Enemy.id_counter += 1

def generate_room():
    description = random.choice(["cave", "passage", "bright_room", "chamber",
"library"])
    items = [random.choice(items_list) for _ in range(random.randint(0, 3))]
    enemies = [Enemy(random.choice(list(enemy_types.keys()))) for _ in
range(random.randint(0, 2))]
    return Room(description, items, enemies)

def render_text(text, y):
    words = text.split(' ')
    lines = []
    current_line = ""

    for word in words:
        if FONT.size(current_line + word)[0] < WIDTH - 40:
            current_line += word + " "
        else:
            lines.append(current_line)
            current_line = word + " "

    lines.append(current_line)

    for i, line in enumerate(lines):
        rendered_text = FONT.render(line, True, WHITE)
        screen.blit(rendered_text, (20, y + i * 40))

def game_loop():
```

```
    player = Player(generate_room())

    while True:
        screen.fill(BLACK)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            elif event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    pygame.quit()
                    sys.exit()
                elif event.key in [pygame.K_e, pygame.K_w, pygame.K_n, pygame.K_s]:
                    # Check if there are any enemies in the room
                    if player.current_room.enemies:
                        total_enemy_health = sum(enemy.stats["health"] for enemy in
player.current_room.enemies)
                        damage = 2 * total_enemy_health
                        player.health -= damage
                        render_text(f"You took {damage} damage from trying to escape!",
400)
                        if player.health <= 0:
                            pygame.quit()
                            sys.exit()
                    else:
                        if event.key == pygame.K_e and "east" in
connections[player.current_room.description]:
                            new_room = generate_room()
                            player.move_to(new_room)
                        elif event.key == pygame.K_w and "west" in
connections[player.current_room.description]:
                            new_room = generate_room()
                            player.move_to(new_room)
                        elif event.key == pygame.K_n and "north" in
connections[player.current_room.description]:
                            new_room = generate_room()
                            player.move_to(new_room)
                        elif event.key == pygame.K_s and "south" in
connections[player.current_room.description]:
                            new_room = generate_room()
```

```
                    player.move_to(new_room)
            elif event.key == pygame.K_p:
                if player.current_room.items:
                    player.pick_up(player.current_room.items[0])
            elif event.key == pygame.K_f:
                if player.current_room.enemies:
                    enemy_id = player.current_room.enemies[0].id
                    result = player.fight_enemy(enemy_id)
                    if result == "Enemy defeated!":
                        player.current_room.enemies = [enemy for enemy in
player.current_room.enemies if enemy.id != enemy_id]
                        if player.health <= 0:
                            pygame.quit()
                            sys.exit()

    render_text(player.current_room.description, 20)

    if player.current_room.items:
        render_text("You see: " + ", ".join(player.current_room.items), 140)

    if player.current_room.enemies:
        for i, enemy in enumerate(player.current_room.enemies):
            enemy_text = f"{enemy.type}: Health: {enemy.stats['health']}"
            render_text(enemy_text, 180 + i * 40)

    def render_multiline_text(text, x, y, line_spacing=10):
        words = text.split(' ')
        lines = []
        current_line = ""
        total_height = 0

        for word in words:
            if FONT.size(current_line + word)[0] < WIDTH - 40:
                current_line += word + " "
            else:
                lines.append(current_line)
                current_line = word + " "

        lines.append(current_line)

        for i, line in enumerate(lines):
```

```
            rendered_text = FONT.render(line, True, WHITE)
            screen.blit(rendered_text, (x, y + i * (FONT.get_height() + line_spacing)))
            total_height += FONT.get_height() + line_spacing

        return total_height


    # Display player's health, inventory and score
    render_text(f"Health: {player.health}", 480)
    inventory_text = "Inventory: " + ", ".join(player.inventory)
    inventory_height = render_multiline_text(inventory_text, 20, 520)

    render_text(f"Items found: {len(player.inventory)}", 520 + inventory_height)

    pygame.display.flip()
    pygame.time.Clock().tick(30)

    if player.health <= 0:
        pygame.quit()
        sys.exit()


connections = {
    "cave": ["passage", "bright_room", "chamber", "library", "west", "south", "east",
"north"],
    "passage": ["bright_room", "cave", "chamber", "library", "west", "south", "east",
"north"],
    "bright_room": ["passage", "cave", "chamber", "library", "west", "south", "east",
"north"],
    "chamber": ["passage", "bright_room", "cave", "library", "west", "south", "east",
"north"],
    "library": ["passage", "bright_room", "cave", "chamber", "west", "south", "east",
"north"]
}

game_loop()
```

3.2 Розробка в деталях

Імпорт бібліотек та ініціалізація гри:

```
import pygame
import sys
import random

pygame.init()

WIDTH, HEIGHT = 800, 800
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Text-based Metroidvania")

WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
FONT = pygame.font.Font(None, 36)
```

Код імпортує бібліотеки pygame і sys, необхідні для роботи гри, а також модуль random.
Задається розмір вікна гри (ширина і висота) і створюється поверхня screen, на якій буде відображатися гра.
Задаються кольори WHITE і BLACK для подальшого використання.
Задається шрифт тексту FONT з розміром 36.

Класи Room та Player:

```
class Room:
    def __init__(self, description, items, enemies):
        self.description = description
        self.items = items
        self.enemies = enemies

class Player:
    def __init__(self, current_room):
        self.current_room = current_room
        self.inventory = []
        self.health = 200
        self.explored_rooms = {current_room.description}

    def move_to(self, new_room):
        self.current_room = new_room
        self.explored_rooms.add(new_room.description)

    def pick_up(self, item):
        if item in self.current_room.items:
            self.current_room.items.remove(item)
            self.inventory.append(item)

    def fight_enemy(self, enemy_id):
```

Клас Room представляє окрему кімнату гри, яка містить опис, список предметів та список ворогів.

Клас Player представляє гравця, який має поточну кімнату, інвентар, здоров'я та список вже досліджених кімнат.

У методі move_to гравець переміщується до нової кімнати.

У методі pick_up гравець забирає предмет з поточної кімнати.

У методі fight_enemy гравець бореться з ворогами у кімнаті.

Ініціалізація гри та головний цикл гри:

```
def generate_room():
    ...

def render_text(text, y):
    ...

def game_loop():
    ...

connections = {
    ...
}

game_loop()
```

generate_room() генерує нову кімнату з випадковим описом, предметами та ворогами.

render_text() відображає текст на екрані гри.

game_loop() представляє головний цикл гри, де гравець може переміщатися, збирати предмети, боротися з ворогами та взаємодіяти з іншими елементами гри.

connections - словник, який визначає доступні напрямки переміщення гравця між кімнатами.

3.3 Інтеграція з бібліотекою Pygame

Інтеграція з бібліотекою Pygame відбувається у кількох аспектах цього коду:

Ініціалізація гри та вікна: Початкова ініціалізація гри та створення вікна виконується за допомогою Pygame. Це робиться за допомогою функцій pygame.init() для ініціалізації, pygame.display.set_mode() для створення вікна гри, і pygame.display.set_caption() для задання заголовка вікна.

Відображення тексту на екрані: Pygame використовується для відображення тексту на екрані. Функція render_text() використовує pygame.font.Font для вибору шрифту та рендерингу тексту на поверхні screen.

Обробка подій клавіатури та миші: Pygame дозволяє обробляти різні події, такі як натискання клавіш та натискання кнопок миші. В цьому коді обробляються події pygame.KEYDOWN, щоб обробити натискання клавіші, та pygame.QUIT, щоб виходити з гри, якщо гравець натисне на кнопку закриття вікна.

Анімація та оновлення стану гри: Хоча цей код не містить анімаційних ефектів, Pygame може використовуватися для відтворення анімацій та оновлення графічного вмісту екрану в реальному часі. У цьому коді гра оновлюється та відображається на екрані у головному циклі game_loop().

Графічні ефекти та візуалізація: Pygame дозволяє створювати різноманітні графічні ефекти та візуалізації. У цьому коді використовуються базові графічні можливості Pygame для відображення тексту, але можливості бібліотеки можуть бути розширені для відтворення анімацій, зображення персонажів та інших графічних ефектів.

Отже, цей код використовує можливості бібліотеки Pygame для створення текстової метроїдванії, включаючи відображення тексту на екрані, обробку подій клавіатури та миші, оновлення стану гри та інші аспекти графічного інтерфейсу користувача.

3.4 Зручність ігор, які запускаються через консоль

Такі ігри поєднують у собі захоплюючий ігровий процес, креативний сюжет та мінімалістичну графіку, що дозволяє гравцям зосередитись на геймплеї та уяві. У цій доповіді ми розглянемо основні причини, чому такі ігри важливі та можуть зацікавити багатьох гравців, а також згадаємо про те, як Pythonista може допомогти користувачам смартфонів грати у подібні ігри.

Зручність Формату

Текстові метроїдванії мають декілька значних переваг, що робить їх зручними для різних категорій гравців. По-перше, вони не вимагають потужного апаратного забезпечення, тому можуть бути запущені на будь-якому сучасному комп'ютері або смартфоні. Це особливо актуально в епоху, коли не всі гравці мають доступ до найновішого обладнання. По-друге, такі ігри зазвичай займають мало місця на диску і не потребують великого обсягу оперативної пам'яті.

Формат текстових ігор дозволяє розробникам сконцентруватися на створенні складних сюжетів та механік без необхідності витрачати значні ресурси на розробку графіки. Це особливо важливо для інді-розробників, які часто працюють в обмежених умовах. Гравці, у свою чергу, отримують можливість насолоджуватися глибоким сюжетом та різноманітними ігровими ситуаціями, що робить гру захопливою та цікавою.

Важливість для Розвитку Уяви

Текстові ігри сприяють розвитку уяви гравців. Оскільки більшість дій описані текстом, гравці самі візуалізують те, що відбувається, створюючи в своїй свідомості унікальні образи персонажів, монстрів та підземель. Це сприяє глибшому зануренню в ігровий процес і дозволяє кожному гравцеві мати унікальний досвід.

Така взаємодія з грою нагадує читання книги, де уява читача відіграє ключову роль у сприйнятті сюжету. Це робить текстові метроїдванії чудовим інструментом для тих, хто цінує літературні аспекти в іграх і любить занурюватися в детально опрацьовані світи.

Гнучкість та Простота Розробки на Python

Python є однією з найпопулярніших мов програмування завдяки своїй простоті та гнучкості. Використання Pygame для створення текстових ігор робить цей процес доступним навіть для початківців у програмуванні. Завдяки цьому зростає кількість розробників, які можуть створювати та експериментувати з іграми, вносячи свій вклад у розвиток жанру.

Платформа Pythonista, зокрема, відкриває нові можливості для розробників, дозволяючи створювати та запускати Python-програми безпосередньо на iOS-пристроях. Це значно спрощує процес тестування та розповсюдження ігор серед користувачів смартфонів. Завдяки Pythonista, розробники можуть швидко перевіряти свої ідеї та вносити зміни, що сприяє більш швидкому випуску якісних ігрових продуктів.

Доступність для Широкої Аудиторії

Текстові метроїдванії привертають увагу різних категорій гравців. Вони підходять як для тих, хто шукає нові виклики та складні завдання, так і для тих, хто хоче просто розслабитися після робочого дня, поринувши в цікавий сюжет. Відсутність складної графіки робить ці ігри доступними для людей з вадами зору, оскільки текст можна легко адаптувати під різні потреби, використовуючи екранні рідери або змінюючи розмір шрифту.

Можливості для Експериментів

Один з найважливіших аспектів текстових метроїдваній - це можливість для експериментів. Розробники можуть легко додавати нові елементи гри, змінювати механіки та вводити нові сюжетні повороти. Це робить процес створення гри динамічним та цікавим, як для самих розробників, так і для гравців, які отримують можливість випробувати нові та незвичні ігрові концепції.

Інді-розробники особливо цінують таку можливість, оскільки вона дозволяє їм втілювати свої творчі ідеї без значних фінансових витрат. Таким чином, текстові метроїдванії стають майданчиком для інновацій, де народжуються нові ігрові жанри та тенденції.

Підсумок

Текстові метроїдванії, створені на Python за допомогою Pygame, є важливим і цікавим явищем в ігровій індустрії. Вони пропонують зручний формат, сприяють розвитку уяви, забезпечують гнучкість розробки та доступність для широкої аудиторії. Завдяки платформі Pythonista, такі ігри можуть бути легко

адаптовані для смартфонів, що відкриває нові можливості для розробників та гравців.

Ці ігри сприяють розвитку інді-сцени та надають можливість експериментувати з новими ідеями та концепціями. Саме тому текстові метроїдванії мають велике значення і можуть зацікавити багатьох людей, незалежно від їх ігрового досвіду та уподобань.

Висновок

Звісно, ось довгий висновок для дипломної роботи, який охоплює усю надану інформацію:

У цій дипломній роботі ми детально розглянули та проаналізували різні аспекти створення текстових відеоігор та їх поєднання з жанром metroidvania. Почавши з розбору основних концепцій metroidvania і їх впливу на індустрію відеоігор, ми дослідили ключові елементи цього жанру, такі як неелінійність, дослідження, прогресія через навички та предмети, і бойова система.

Далі ми розглянули текстові відеоігри та їх особливості, включаючи інтерактивні історії та ігри жанру "interactive fiction". Порівнявши їх з жанром metroidvania, ми з'ясували, що обидва жанри мають свої унікальні особливості, такі як неелінійність сюжету та дій, а також можливості вибору гравця.

Поєднання жанрів metroidvania та текстових відеоігор виявилося цікавим напрямком для розвитку індустрії відеоігор. Це поєднання дозволяє створювати ігри з глибоким сюжетом, багатими можливостями дослідження та унікальним геймплеєм. Гравці можуть насолоджуватися поєднанням розвідки великих світів з різноманітними завданнями та можливістю впливати на розвиток сюжету.

Наш аналіз також включав практичний приклад створення текстової metroidvania гри за допомогою бібліотеки Pygame у мові програмування Python. Ми розглянули основні частини коду, такі як створення кімнат, рух гравця, боротьба з ворогами та взаємодія з предметами.

Загалом, дослідження жанрів metroidvania та текстових відеоігор, а також їх поєднання, надає великий потенціал для творчості та інновацій в індустрії відеоігор. Ці напрямки варто дослідити далі для створення захоплюючих та неповторних геймплейних досвідів для гравців усього світу.

Список посилань

Jason McIntosh, "A brief history of text-based games and open source", opensource.com, 2018

"A Guide to the World of Metroidvanias", plarium.com, 2023

"Python Game Development Tutorials", realpython.com

nikhilaggarwal3, "Pygame Basics", GeeksforGeeks, 2024

Pygame.org Docs

"Pygame Tutorial", TutorialsPoint

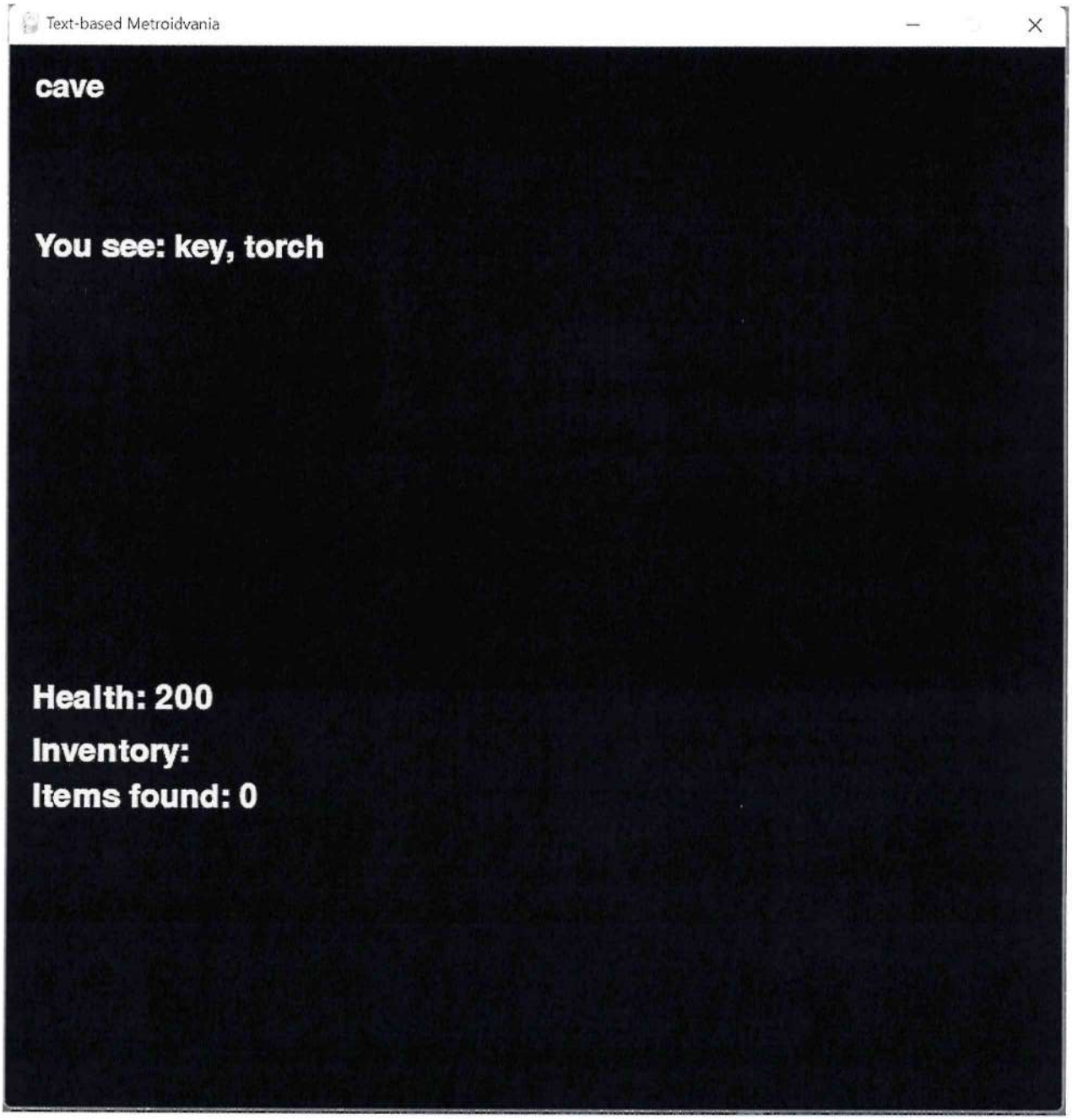
Josh Bycer, "The Foundation of Metroidvania Design", gamedeveloper.com, 2017

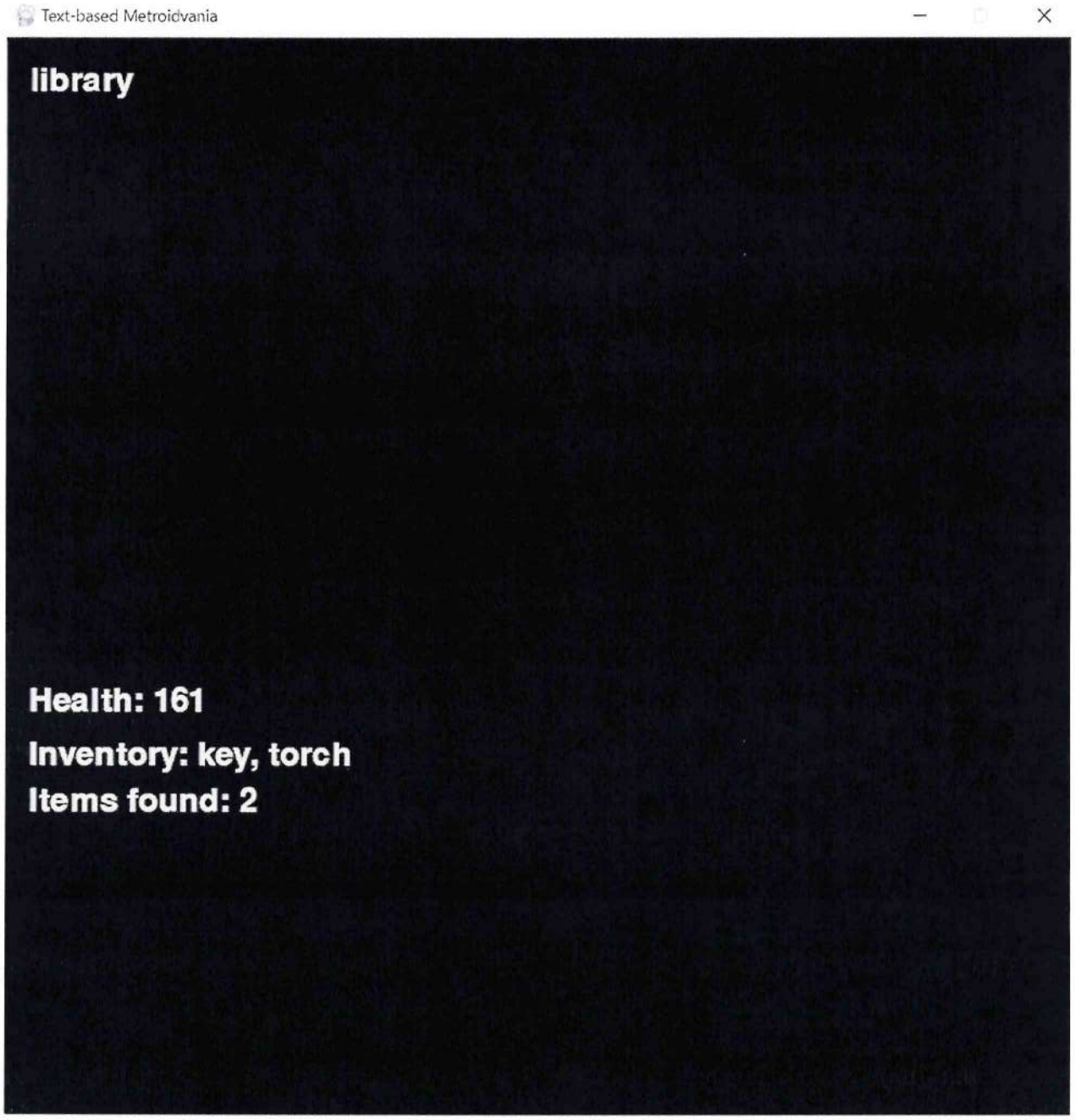
Kevin Purdy, "50 Years of Text Games parses the rich history of a foundational genre", arstechnica.com, 2023

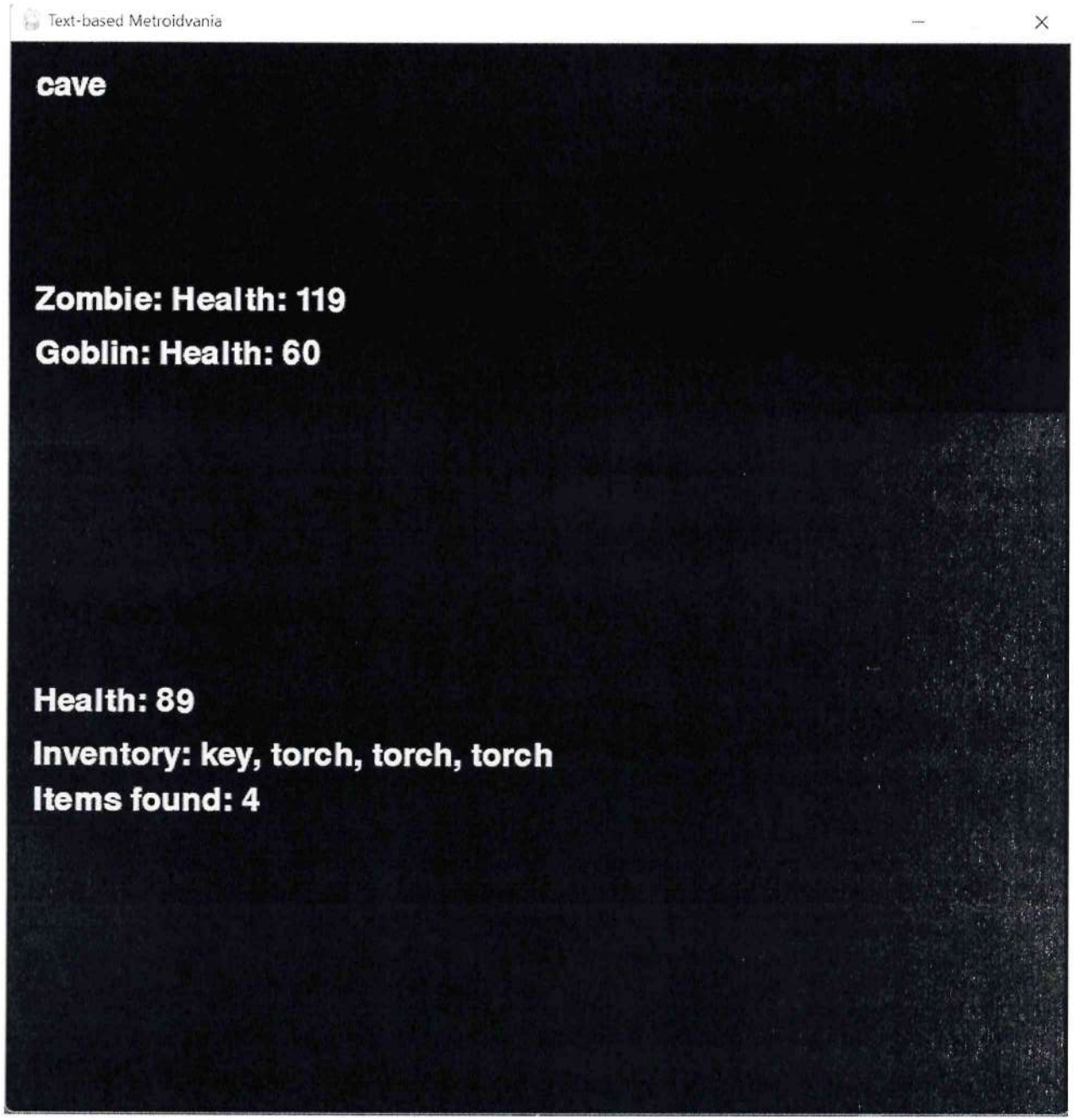
Dominik Landwehr, "Back to the early days of computer gaming with text adventures", blog.nationalmuseum.ch, 2020

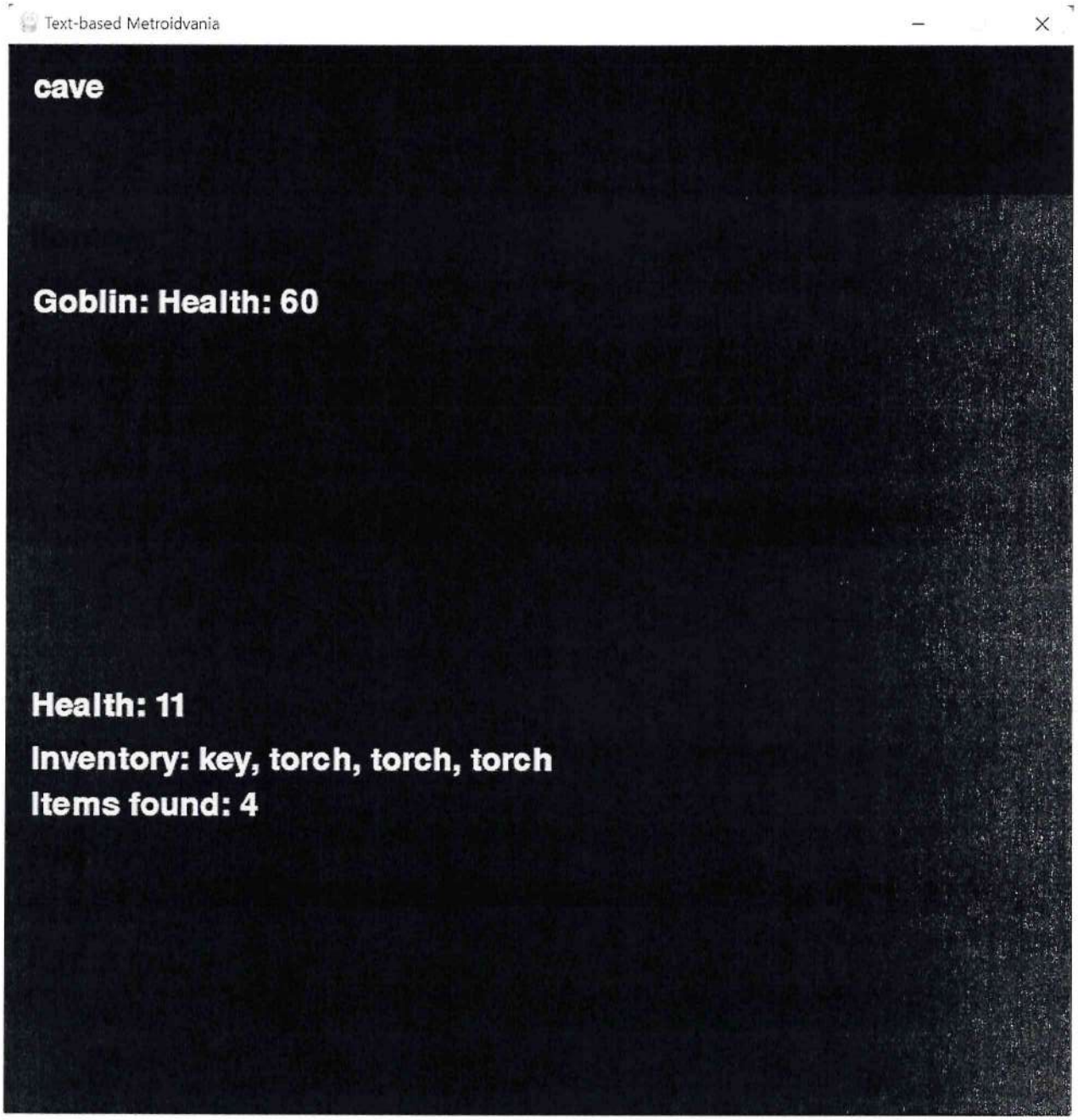
Додатки

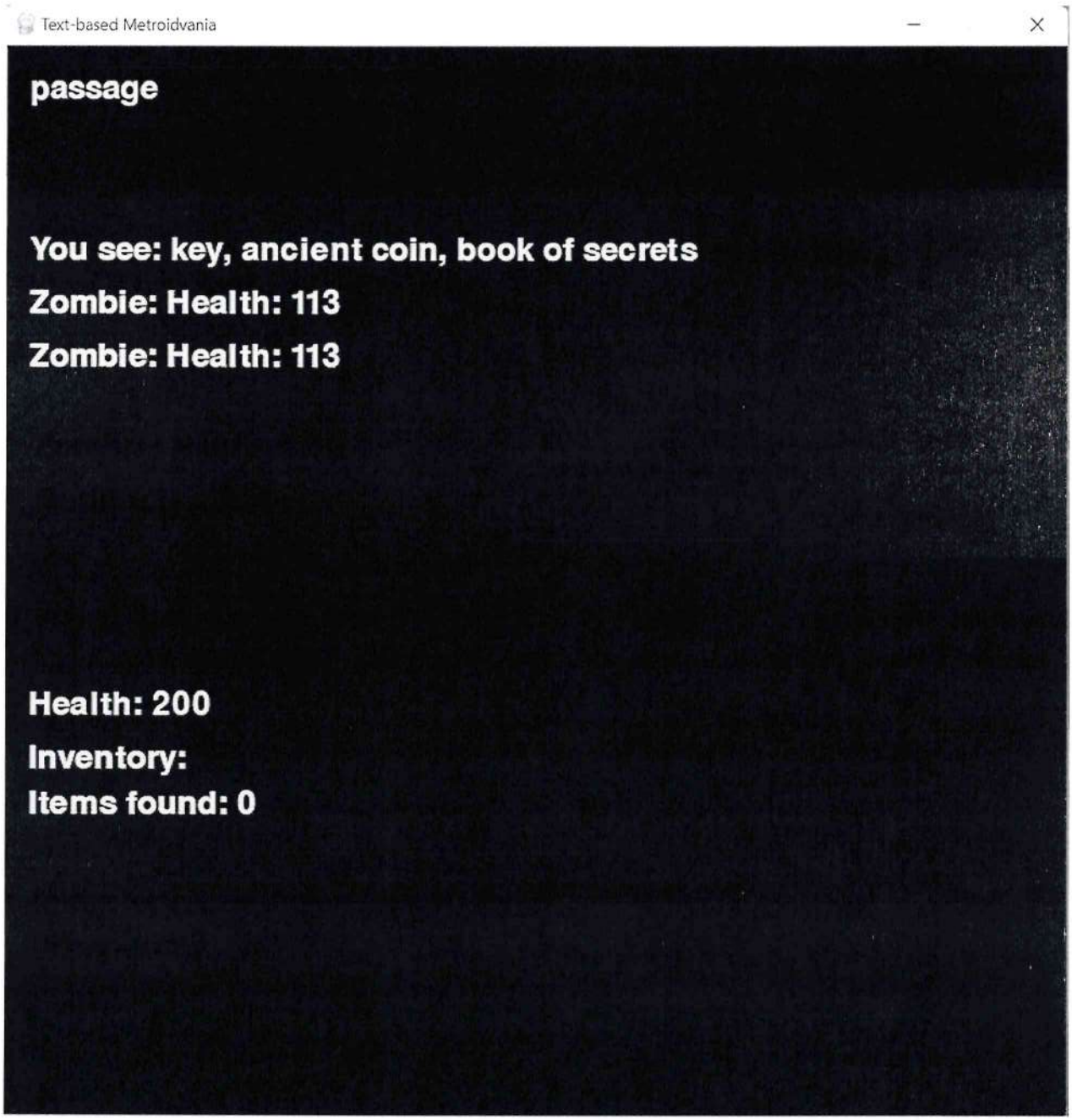
Додаток А

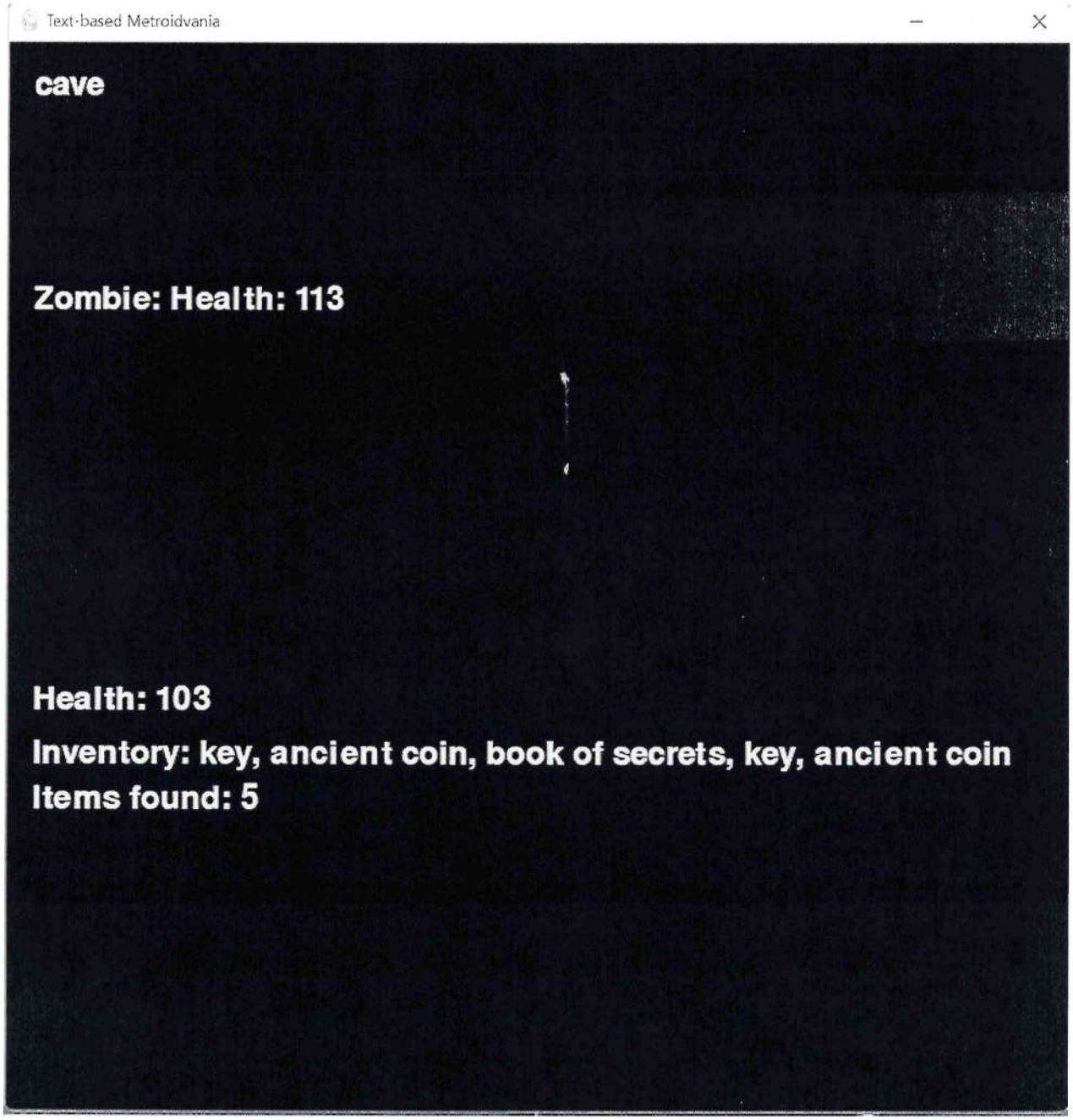


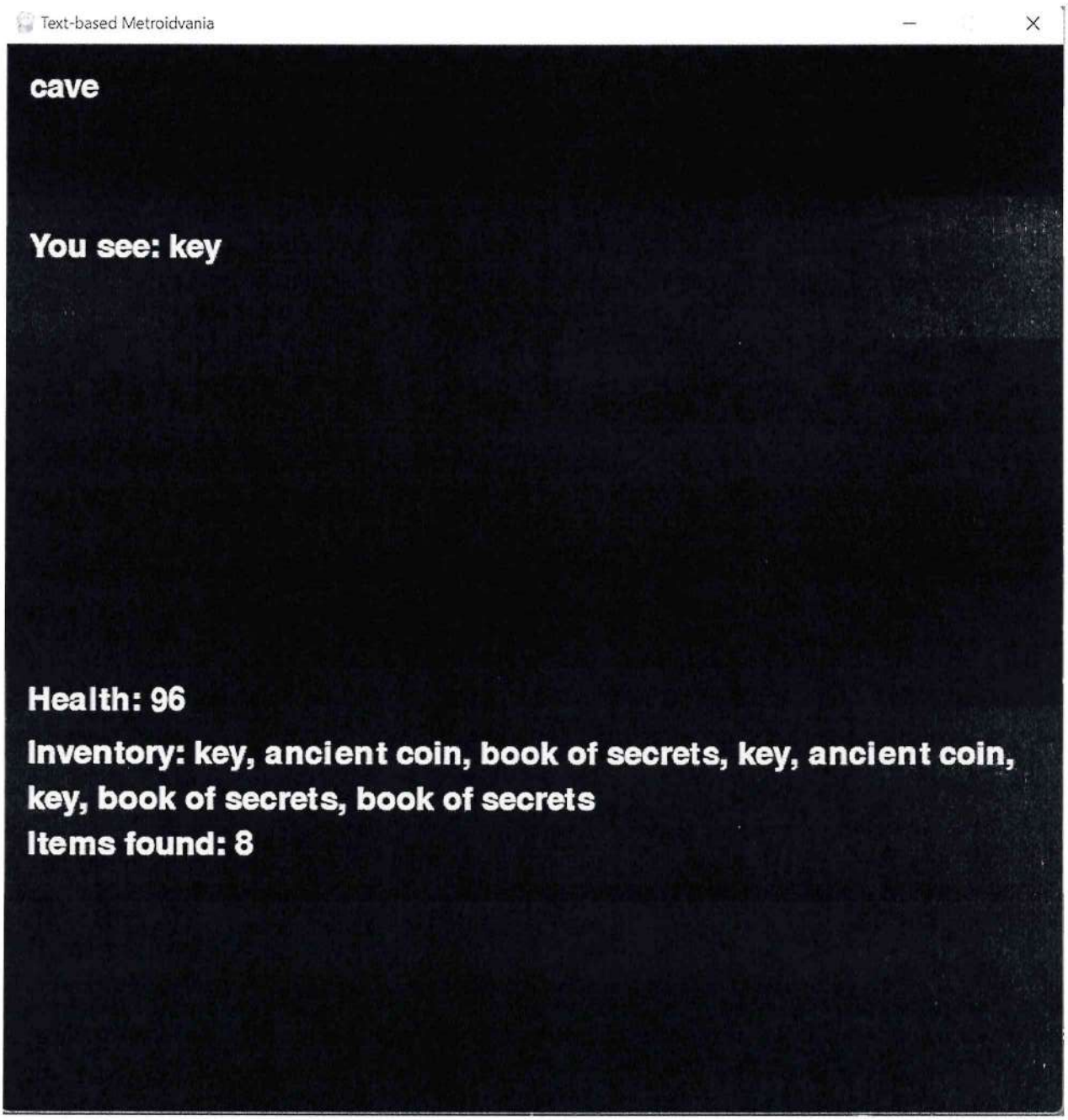

Text-based Metroidvania
library
Health: 161
Inventory: key, torch
Items found: 2

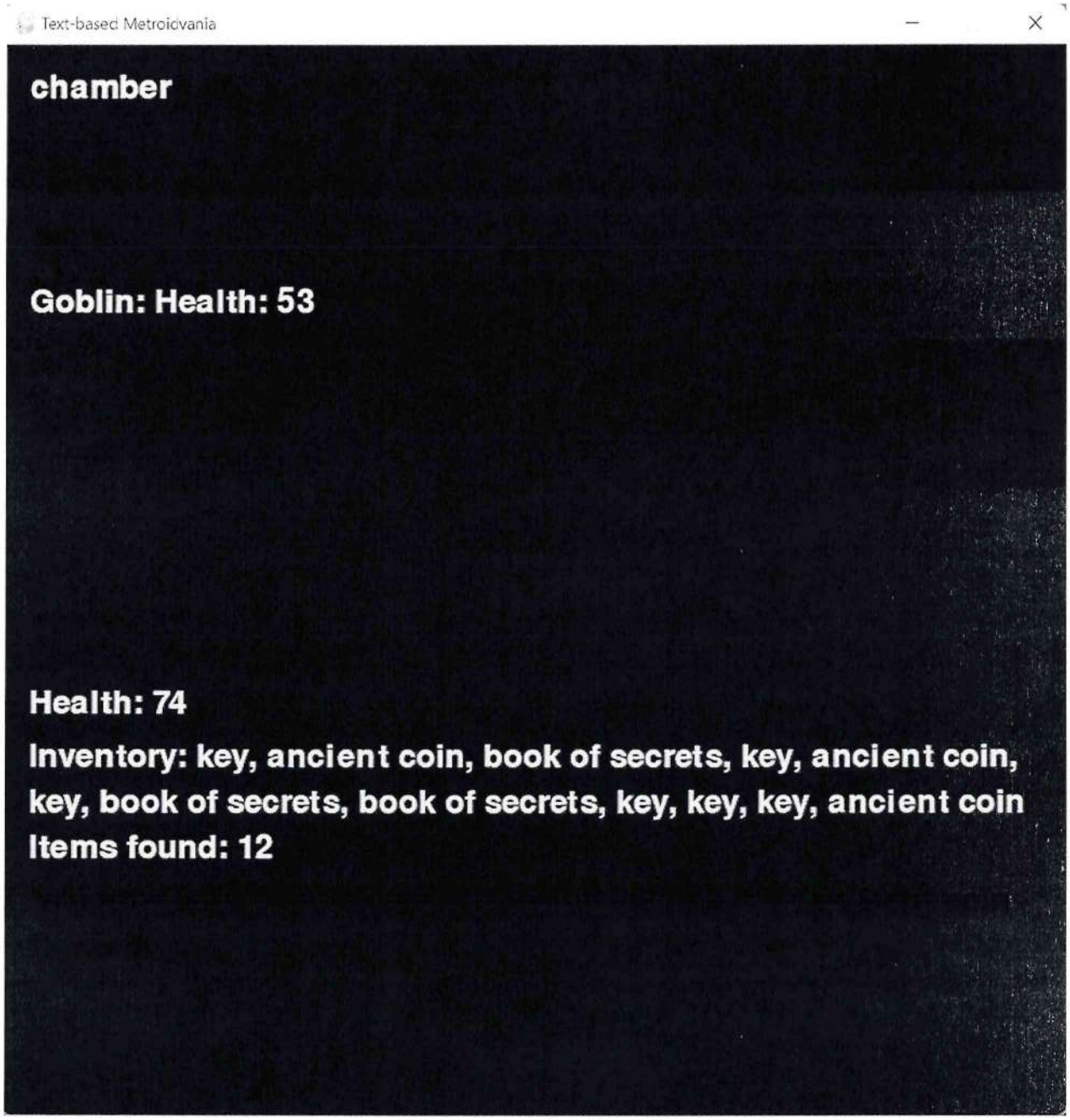

Text-based Metroidvania
cave
Zombie: Health: 119
Goblin: Health: 60
Health: 89
Inventory: key, torch, torch, torch
Items found: 4

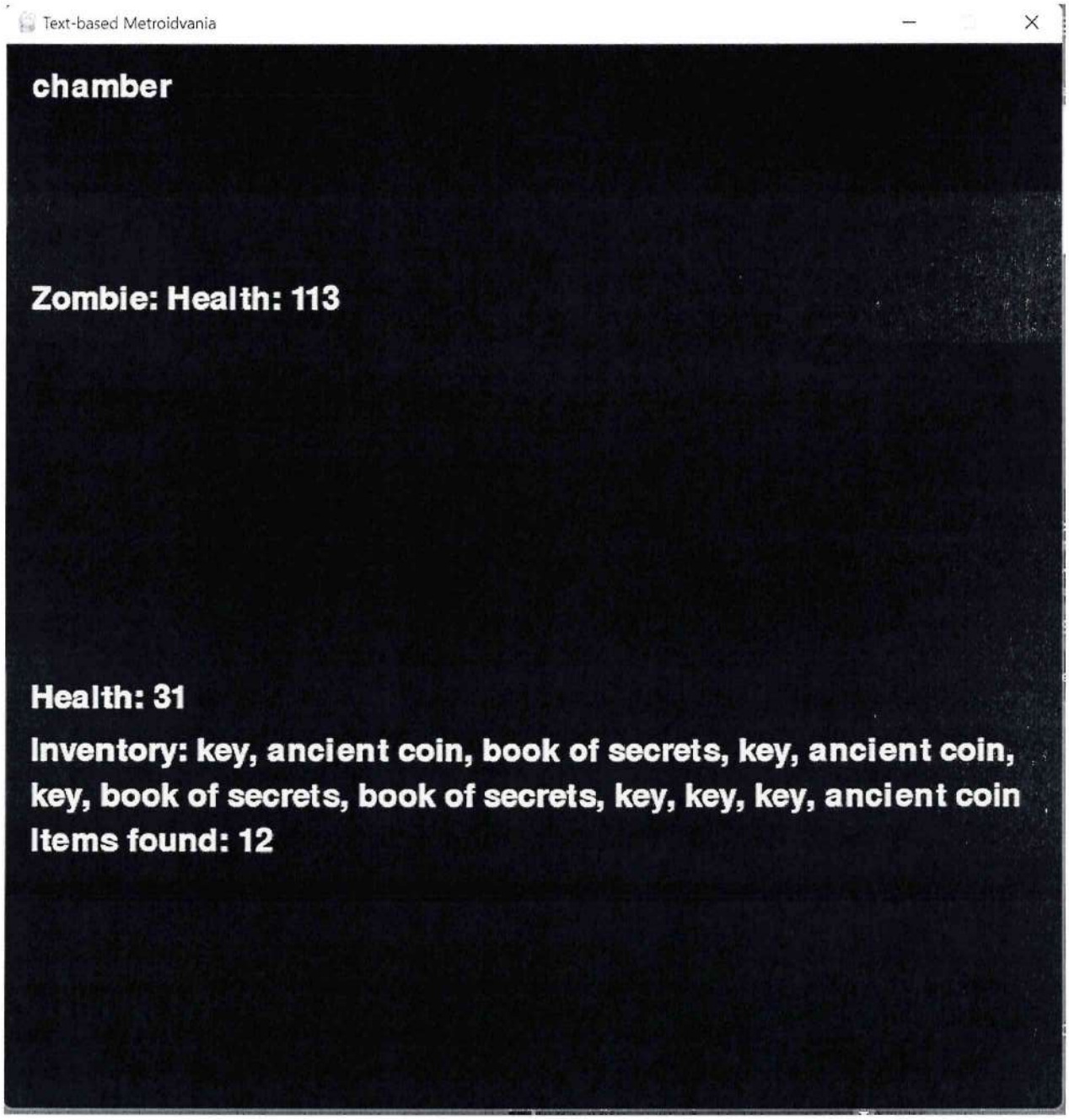

Text-based Metroidvania
cave
Goblin: Health: 60
Health: 11
Inventory: key, torch, torch, torch
Items found: 4

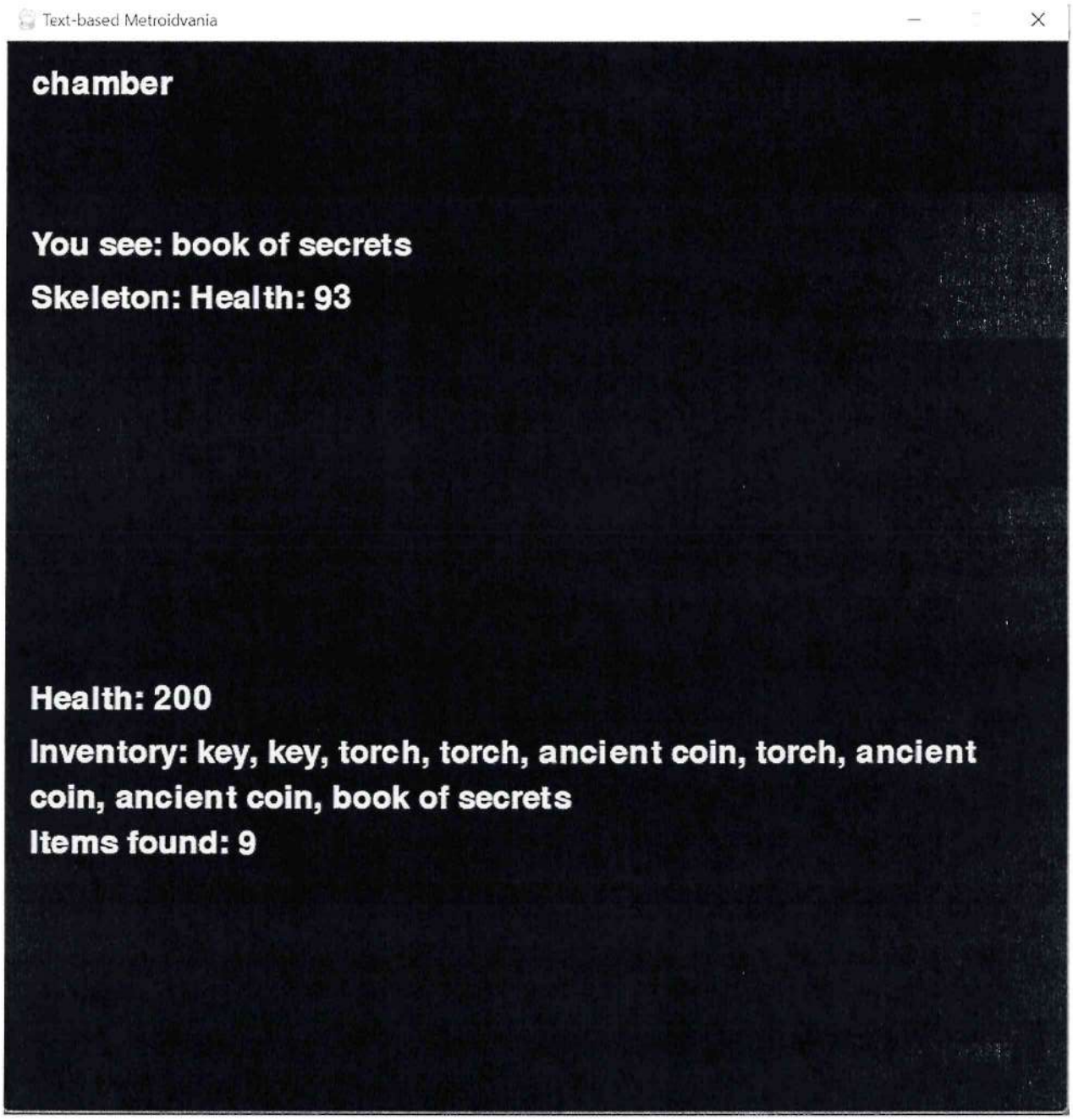

Text-based Metroidvania
passage
You see: key, ancient coin, book of secrets
Zombie: Health: 113
Zombie: Health: 113
Health: 200
Inventory:
Items found: 0

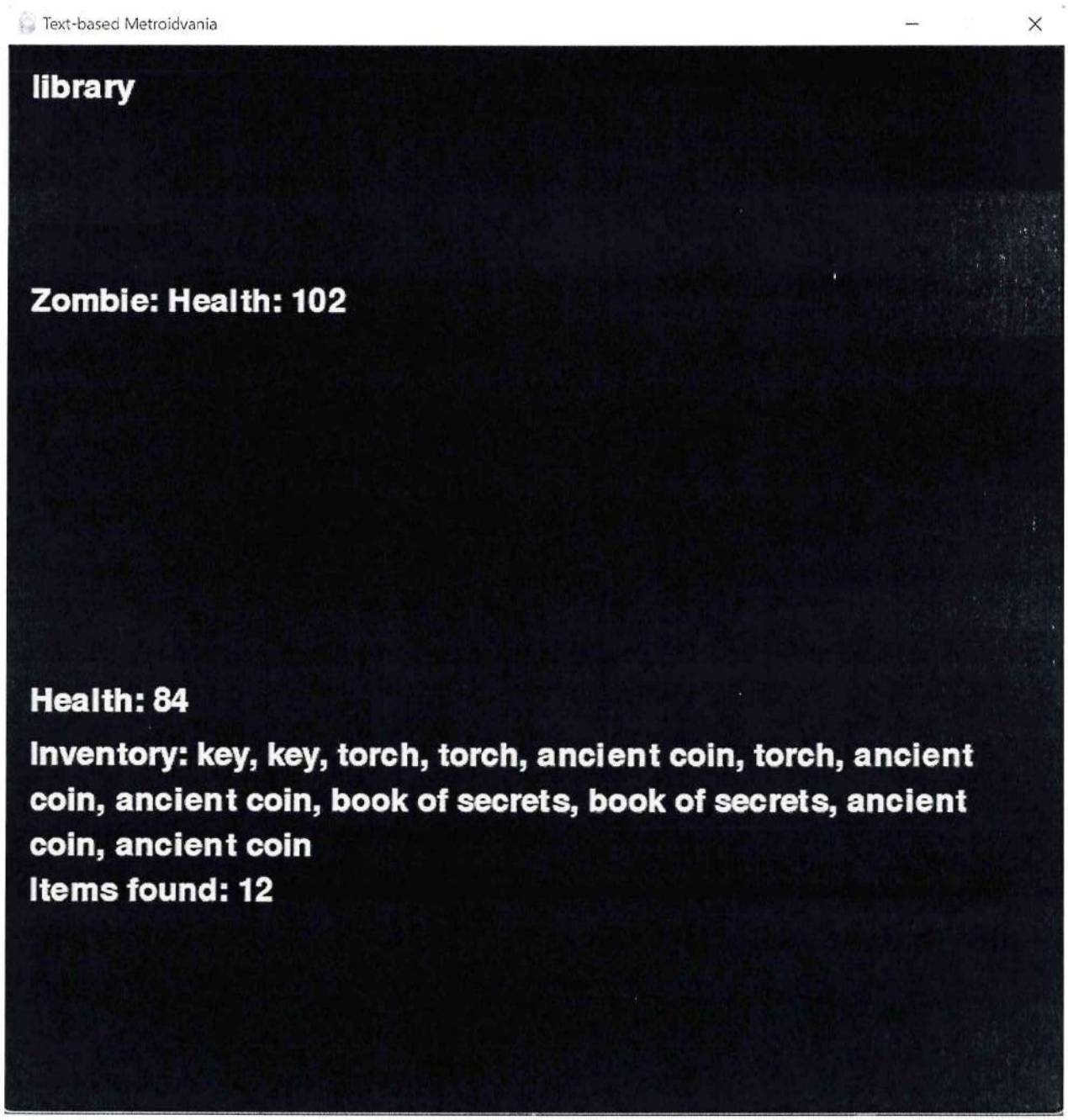

Text-based Metroidvania
cave
Zombie: Health: 113
Health: 103
Inventory: key, ancient coin, book of secrets, key, ancient coin
Items found: 5

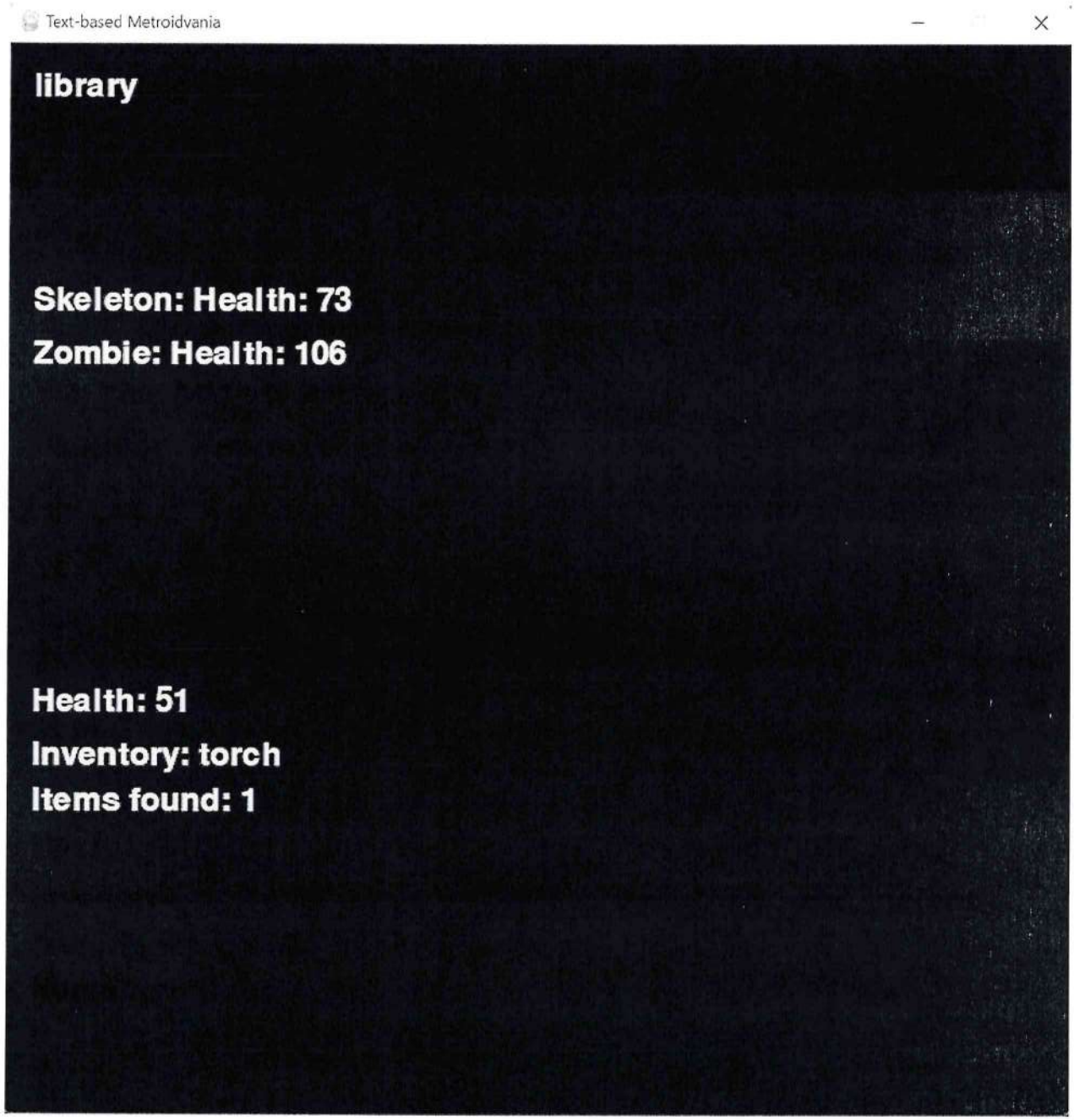

Text-based Metroidvania
cave
You see: key
Health: 96
Inventory: key, ancient coin, book of secrets, key, ancient coin, key, book of secrets, book of secrets
Items found: 8

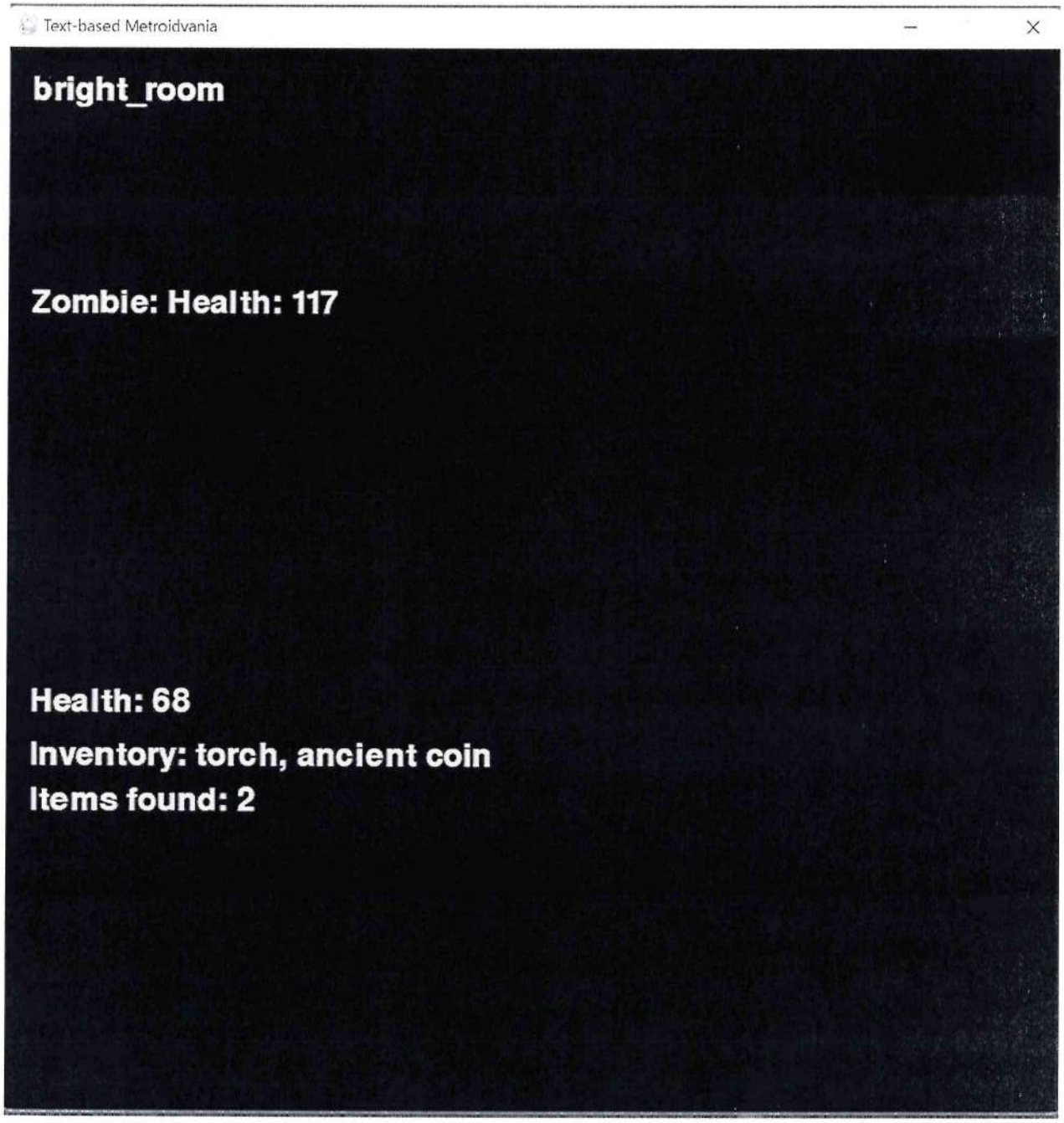

Text-based Metroidvania
chamber
Goblin: Health: 53
Health: 74
Inventory: key, ancient coin, book of secrets, key, ancient coin, key, book of secrets, book of secrets, key, key, key, ancient coin
Items found: 12


Text-based Metroidvania
chamber
Zombie: Health: 113
Health: 31
Inventory: key, ancient coin, book of secrets, key, ancient coin, key, book of secrets, book of secrets, key, key, key, ancient coin
Items found: 12


Text-based Metroidvania
chamber
You see: book of secrets
Skeleton: Health: 93
Health: 200
Inventory: key, key, torch, torch, ancient coin, torch, ancient coin, ancient coin, book of secrets
Items found: 9


Text-based Metroidvania
library
Zombie: Health: 102
Health: 84
Inventory: key, key, torch, torch, ancient coin, torch, ancient coin, ancient coin, book of secrets, book of secrets, ancient coin, ancient coin
Items found: 12


Text-based Metroidvania
library
Skeleton: Health: 73
Zombie: Health: 106
Health: 51
Inventory: torch
Items found: 1


Text-based Metroidvania
bright_room
Zombie: Health: 117
Health: 68
Inventory: torch, ancient coin
Items found: 2