1. Pojęcie silnika gry

Silnik gry to zespół narzędzi i bibliotek służący do tworzenia gier komputerowych. Najważniejsze jego elementy to:

- Środowisko programistyczne (np. Unity, Unreal Engine)
- Biblioteki graficzne i dźwiękowe
- System zarządzania zasobami
- Wspomaganie fizyki
- Osiągnięcia i postępy gracza
- Interfejs użytkownika

2. Pojęcie ludologii

Ludologia to interdyscyplinarna nauka badająca media i komunikację. W kontekście gier:

- Badanie wpływu gier na społeczeństwo
- Analiza treści i narracji gier
- Studia nad graczami i ich zachowaniami
- Teoria gier jako medium

3. Koncepcja sceny w Unity

Scena w Unity to najniższy poziom hierarchii obiektów. Kluczowe elementy:

- Obiekty (3D, 2D, UI)
- Komponenty obiektów
- Transformacja (pozycja, skala, rotacja)
- Ruch i fizyka
- Interakcje między obiektami

4. Różne widoki w grach

Widoki w grach to różne perspektywy na świat gry:

- Widok pierwszoosobowy (FPS)
- Widok trzeci osobowy
- Widok izometryczny
- Widok perspektywiczny

- Widok z góry (top-down)
- Widok poziomy (side-scrolling)

5. Ogólna koncepcja modelowania fizyki

Modelowanie fizyki w grach obejmuje:

- Ruch obiektów
- Interakcje między obiektami
- Siły i przyspieszenia
- Kolidery i mechanizmy kolizji
- Fizyka rzeczywista vs symulowana

6. Różne tryby ustawień AddForce/AddTorque

Tryby ustawień AddForce/AddTorque to różne sposoby aplikowania sił:

- Tryb globalny - na cały obiekt
- Tryb lokalny - na konkretne komponenty
- Tryb odliczenia - wyłączenie efektów
- Tryb ciągłej aktualizacji

7. Podstawy implementacji fizyki w Unity

Implementacja fizyki w Unity obejmuje:

- Ustawienie zmiennych fizycznych obiektom
- Konfiguracja kolidera
- Używanie skryptów do kontroli ruchu
- Optymalizacja wydajności

8. Podstawowe mechanizmy działania skryptów

Mechanizmy działania skryptów w Unity:

- Cykl życia metod (Awake, Start, Update itp.)
- Używanie tagów i warunków
- Interakcje między obiektami
- Obsługa zdarzeń

9. Różnice między klasami a strukturami w Unity/C#

Klasy vs struktury:

- Klastry są abstrakcyjne, struktury konkretne
- Struktury mają mniejszą wydajność
- Klastry mogą dziedziczyć po sobie
- Struktury nie mogą dziedziczyć

10. Podejście do serializacji i hermetyzacji w Unity

Serializacja i hermetyzacja:

- Serializacja - konwersja danych na format pliku
- Hermetyzacja - ochrona danych przed edycją
- Ustawienie flag serializacji
- Użycie atrybutów [SerializeField]

11. Elementy Game Design Document

Game Design Document powinien zawierać:

- Opis gry i jej mechaniki
- Celowe grupy docelowej
- Plan rozwoju i budżet
- Przedstawione koncepcje artystyczne
- Szacunkowy czas produkcji

12. Etapy produkcji gier

Etapy produkcji gier:

- Faza planowania
- Faza projektowania
- Faza implementacji
- Faza testowania
- Faza optymalizacji
- Faza publikacji

13. Koncepcja tetrady podstawowej

Tetraada podstawowa to zestaw najprostszych elementów:

- Obiekty
- Komponenty
- Siły i przyspieszenia
- Interakcje
- Systemy kolizji

14. Koncepcja tetrady warstwowej

Tetraada warstwowa dzieli grę na:

- Warstwa modelu
- Warstwa widoku
- Warstwa sterowania
- Warstwa fizyki
- Warstwa audio

15. Podstawowe komponenty w Unity

Podstawowe komponenty w Unity:

- Transform - pozycja, skala, rotacja
- Renderer - renderowanie grafiki
- Collider - detekcja kolizji
- BoxCollider - prostokątny collider
- Rigidbody - fizyka obiektu
- Animator - animacje

16. Podstawowe obiekty gry w Unity

Podstawowe obiekty gry w Unity:

- GameObject - podstawowy obiekt
- Prefab - uogólniony obiekt
- Scene - scena gry

- Asset - zasób graficzny
- ScriptableObject - dane uniwersalne

17. Koncepcja tworzenia UI w Unity

Tworzenie UI w Unity:

- Używanie Canvas systemu
- Dodawanie UI elements (text, buttons, images)
- Konfiguracja layout grup
- Ustawianie efektów interakcji
- Lokalizacja tekstów

18. Pojęcie gry

Gra to interaktywna forma rozrywki:

- Zawiera elementy zabawy i wyzwania
- Posługuje się grafiką i dźwiękiem
- Pozwala na interakcję gracza
- Ma celowe grupy docelowej

19. Ogólna koncepcja modelowania oświetlenia w grafice komputerowej

Modelowanie oświetlenia obejmuje:

- Źródła światła (punktowe, kierunkowe, opuszkowe)
- Jaskrawość i jasność
- Kolor i temperatura barwy
- Odbicie i refleksy
- Stosunki między obiektami a światłem

20. Koncepcja światła w Unity

Światło w Unity:

- Źródła światła (Directional Light, Point Light itp.)
- Reflektory i odbijacze
- Stosunki jasności i jaskrawości

- Efekty atmosferyczne
- Śledzenie oświetlenia

21. Podstawy tworzenia animacji w Unity

Tworzenie animacji w Unity:

- Używanie Animator Controller
- Tworzenie statków animacyjnych
- Dodawanie klipów animacyjnych
- Konfiguracja przejść między stanami
- Osiągnięcia i postępy gracza

22. Różnice między klipem animacyjnym a Animatorem w Unity

Klip animacyjny vs Animator:

- Klip - pojedynczy ruch lub sekwencja
- Animator - zarządzanie różnymi stanami
- Klipy łatwiejsze w użyciu, Animator bardziej elastyczny
- Animator umożliwia bardziej zaawansowane efekty

23. Koncepcja kamery w Unity

Kamera w Unity:

- Pozycjonowanie i kontrola perspektywy
- Ustawienia ostrości i czułości
- Sterowanie ruchem
- Włączenie efektów chmur i atari
- Zapisywanie i wczykiwanie pozycji

24. Mechanizmy obsługi wejść i kontrolerów

Obsługa wejść i kontrolerów:

- Detekcja wejść (keyboard, gamepad, mouse)
- Mapowanie funkcji na kontrole
- Obsługa wieloosobowego gry

- Adaptacja interfejsu dla różnych urządzeń
- Testowanie na rzeczywistych sprzętach

25. Różnica między rzutowaniem perspektywicznym a ortograficznym (ortogonalnym)

Rzut perspektywiczny vs ortograficzny:

- Perspektywiczny - odzwierciedla rzeczywistość
- Ortograficzny - prosty, bez głębokości
- Użycie w różnych typach gier i widoków
- Efekty specjalne w obu przypadkach

26. Jakimi elementami wyróżnia się gra od innych form rozrywki

Elementy wyróżniające grę:

- Interaktywność i udział gracza
- Struktura zakończeń i postępów
- Celowe grupy docelowej
- Elementy wyzwaniowych i nagradzających
- Długotrwałość i powtarzalność

27. ScriptableObject

ScriptableObject to:

- Specjalny typ danych w Unity
- Nie jest obiektem sceny
- Można go edytować w Edytorze Zasobów
- Idealny do przechowywania uniwersalnych danych
- Łatwy do użycia w wielu miejscach kodu

28. Różne sposoby zapisów postępów w grach przygotowanych za pomocą Unity

Metody zapisywania postępów:

- Save System - własny system zapisu
- PlayerPrefs - proste dane użytkownika
- JSON czy XML pliki

- Bazy danych (SQLite itp.)
- Cloud saving (Unity Analytics)

29. Modelowanie komponentu transform (bez wzorów)

Komponent Transform w Unity:

- Pozycja (Position)
- Skala (Scale)
- Rotacja (Rotation)
- Lokalizacja w przestrzeni
- Transformacji jako zmienne
- Metody transformacji

30. Atrybuty w Unity używane w skryptach

Atrybuty w Unity:

- [SerializeField] - prywatne pole serializowane
- [System.Serializable] - klasa serializowana
- [RequireComponent] - konieczność posiadania komponentu
- [ExecuteAlways] - wykonanie w Ed

31. Standardowy cykl życia metod w Unity

Standardowy cykl życia metod w Unity obejmuje następujące fazę:

1. Awake() - wywoływana gdy obiekt zostaje aktywowany, zawsze przed Start()
2. OnEnable() - wywoływana gdy skrypt staje się aktywny, zawsze po Awake()
3. Start() - wywoływana przed pierwszą aktualizacją ramki, jeśli skrypt jest aktywny
4. Update() - wywoływana raz na ramkę, główna funkcja aktualizacji
5. LateUpdate() - wywoływana raz na ramkę, po Update()
6. OnDisable() - wywoływana gdy skrypt staje się nieaktywny
7. OnDestroy() - wywoływana przed usunięciem obiektu z sceny

Kluczowe punkty:

- Awake() i OnEnable() są wywoływane tylko raz przy aktywowaniu obiektu

- Start() jest wywoływany przed Update()
- LateUpdate() jest używany głównie dla kamer i innych elementów, które powinny śledzić ruch gracza

32. Leniwe wyrzucanie w Unity

Leniwe wyrzucanie to mechanizm optymalizacji wydajności w Unity:

- Zmniejsza liczbę aktualizacji ramki
- Używa czasu od poprzedniej ramki jako czasu między aktualizacjami
- Działa na obiekty z Rigidbody lub Collider

Kluczowe punkty:

- Aktywne leniwe wyrzucanie zmienia Time.fixedDeltaTime
- Działa niezależnie od częstotliwości odświeżania ekranu
- Polecane do fizyki i obliczeń niezależnych od FPS

33. Mechanizm kolizji w Unity

Mechanizm kolizji w Unity obejmuje:

1. Collider - komponent określający kształt obiektu dla celów kolizji
   - Przykłady: BoxCollider, SphereCollider
2. Rigidbody - komponent umożliwiający interakcje fizyczne
   - Może być połączony z Colliderem
3. Triggery - specjalny tryb Collidera, który wyzwala zdarzenia bez kolizji
   - Pozwala na interakcje bez uszkodzeń obiektów

Kluczowe metody:

- OnCollisionEnter() - wywoływana przy pierwszej kolizji
- OnCollisionStay() - wywoływana podczas trwającej kolizji
- OnCollisionExit() - wywoływana przy końcu kolizji
- OnTriggerEnter() - wywoływana przy pierwszym przecięciu triggera
- OnTriggerStay()
- OnTriggerExit()

Kluczowe punkty:

- Collider i Rigidbody są niezbędne do interakcji fizycznych
- Triggery pozwalają na bardziej zaawansowane mechaniki gry
- Można konfigurować warunki kolizji w Unity Editorze

Mechanizm kolizji jest kluczowy dla tworzenia interakcji między obiektami w grach Unity, umożliwiając zarządzanie zderzeniami i przepływem ruchu.