Załącznik nr 1

OPIS PROGRAMU DLA KIERUNKU STUDIÓW

Tworzenie gier komputerowych i technologie interaktywne

I stopień

profil praktyczny

1. OGÓLNA CHARAKTERYSTYKA PROGRAMU STUDIÓW	
Wydział prowadzący studia:	**Wydział Transportu i Informatyki**
1.1 Nazwa programu/kierunku studiów/specjalności	Tworzenie gier komputerowych i technologie interaktywne, specjalności do wyboru: 1. Programowanie gier komputerowych, 2. Grafika i projektowanie poziomów w grach komputerowych
1.2 Poziom studiów	Studia pierwszego stopnia
1.3 Poziom Polskiej Ramy Kwalifikacji	6 poziom Polskiej Ramy Kwalifikacji
1.4 Profil studiów	Praktyczny
1.5 Forma /-y studiów	Studia stacjonarne, niestacjonarne
1.6 Tytuł zawodowy nadany absolwentom, KOD ISCED. Opis syntetyczny charakterystyk zawodowych, stanowiska pracy absolwenta po ukończeniu studiów	Inżynier; Kod ISCED: podgrupa technologii teleinformacyjnych 061; Tworzenie i analiza oprogramowania i aplikacji 0613 Absolwent kierunku to inżynier wszechstronnie przygotowany do pracy w branży gier komputerowych i technologii interaktywnych. Dzięki praktycznemu profilowi kształcenia, posiada wiedzę i umiejętności w zakresie programowania gier, grafiki 2D i 3D, projektowania poziomów (level design), znajomości silników gier oraz wykorzystania technologii immersyjnych, takich jak VR i AR. Absolwent jest przygotowany nie tylko do pracy w zespole, ale również do założenia i prowadzenia własnej działalności gospodarczej. Osoba ta potrafi wykorzystać nabyte kompetencje do formułowania i rozwiązywania problemów o charakterze praktycznym obejmujące techniczne aspekty tworzenia gier. Posiada zaawansowaną wiedzę z zakresu programowania mechaniki gry oraz optymalizacji wydajności w popularnych silnikach gier. Tworzy wizualne elementy gier, które są zarówno estetyczne, jak i funkcjonalne. W szczególności potrafi wykonać zadania obejmujące: • proces produkcji gier komputerowych, dobór środowisk programistycznych oraz narzędzi do projektowania, implementacji, testowania i wdrażania gier, w tym na urządzenia mobilne, • projektowanie mechaniki gier, narracji oraz ergonomii rozgrywki, • interakcję człowieka z komputerem oraz ich zastosowanie w technologiach immersyjnych • projektowanie graficzne obejmujące tworzenie koncepcji wizualnych, modeli 2D i 3D postaci, środowisk i przedmiotów; • tworzenie animacji postaci oraz efektów specjalnych; • projektowanie poziomów gier uwzględniając fabułę, narrację, cel gry oraz funkcjonalność przestrzeni; • zasady wydawania gier, współpracy z wydawcami, promocji gry i budowania społeczności graczy Osoba posiadająca ww. kwalifikacje jest przygotowana do pracy w przedsiębiorstwach/ jednostkach o różnym profilu działania, a w szczególności w: • firmach informatycznych specjalizujących się w produkcji gier komputerowych; • firmach marketingowych; • studiach graficznych; • firmach zajmujących się tworzeniem i wdrażaniem różnego rodzaju oprogramowania; • firmach, które zajmują się wykorzystaniem nowoczesnych technologii informatycznych w praktyce, • instytucjach wykorzystujących gry i technologie interaktywne w edukacji, kulturze i biznesie. na stanowiskach: • programisty gier komputerowych; • projektanta poziomów w grach; • grafika 2D/3D; • projektanta interfejsów użytkownika;

	• animatora w grach komputerowych i reklamach; • twórcy aplikacji VR/AR; • specjalisty technologii interaktywnych; • analityka i testera gier; • specjalisty ds. silników gier; przedsiębiorcy w branży gier.
1.7 Liczba semestrów i punktów ECTS konieczna do ukończenia studiów	VII semestrów, 210 ECTS
1.8 Łączna liczba godzin zajęć dydaktycznych na studiach stacjonarnych/niestacjonarnych	2660 - godzin zajęć dydaktycznych na studiach stacjonarnych; 1850 - godzin zajęć dydaktycznych na studiach niestacjonarnych; w tym 6 –cio miesięczne praktyki zawodowe na studiach stacjonarnych i studiach niestacjonarnych.
1.9 Łączna liczba punktów ECTS uzyskana w ramach zajęć prowadzonych z bezpośrednim udziałem nauczycieli akademickich lub innych osób prowadzących zajęcia	106 punktów ECTS z 210 punktów ECTS w programie na studiach stacjonarnych, 74 punkty ECTS z 210 punktów ECTS w programie na studiach niestacjonarnych
1.10 Liczba punktów ECTS w ramach zajęć z dziedziny nauk humanistycznych lub nauk społecznych	12 punktów ECTS

2. OKREŚLONE W PROGRAMIE STUDIÓW EFEKTY UCZENIA SIĘI PRZYPISANIE DYSCYPLIN NAUKOWYCH

2.1 Przypisanie dyscyplin

Dziedzina naukowa: dziedzina nauk inżynieryjno-technicznych

Lp.	Nazwa dyscypliny	Liczba punktów ECTS	%
1.	Informatyka techniczna i telekomunikacja	210	100
Razem liczba ECTS i procent ECTS w programie studiów		210	100

2.2 Kierunkowe efekty uczenia się w odniesieniu do PRK

Nazwa kierunku:	**Tworzenie gier i technologie interaktywne**		
Poziom kształcenia:	**POZIOM 6 PRK - Studia pierwszego stopnia**		
Profil kształcenia:	**Praktyczny**	Odniesienie do:	
Symbol efektów uczenia się dla programu studiów	**Efekty uczenia się po ukończeniu studiów pierwszego stopnia**	uniwersalnych charakterystyk dla danego poziomu PRK	charakterystyk drugiego stopnia efektów uczenia się dla kwalifikacji na poziomach 6–7 PRK: Poziom 6 / Kompetencje inżynierskie
WIEDZA Absolwent zna i rozumie:			
K_W01	w zaawansowanym stopniu kluczowe pojęcia z zakresu dyscypliny naukowej Informatyka techniczna i telekomunikacja niezbędne do: • opisu i analizy algorytmów i struktur danych, • opisu i analizy działania oraz wdrażania gier komputerowych, • opisu i analizy działania technologii immersyjnych oraz praktyczne zastosowanie tej wiedzy w działalności zawodowej związanej z kierunkiem studiów Tworzenie gier komputerowych i technologie interaktywne.	P6U_W	P6S_WG
K_W02	w zaawansowanym stopniu zagadnienia w zakresie projektowania oraz produkcji gier komputerowych i aplikacji interaktywnych od fazy koncepcyjnej po finalne wdrożenie i testowanie	P6U_W	P6S_WG
K_W03	w zaawansowanym stopniu wybrane zagadnienia dotyczące technicznych i matematycznych podstaw informatyki, a także jest zdolny do wykorzystania tej wiedzy w działalności zawodowej związanej z kierunkiem studiów.	P6U_W	P6S_WG
K_W04	w zaawansowanym stopniu zagadnienia i terminologię w zakresie architektury komputerów, urządzeń peryferyjnych oraz urządzeń sieciowych wykorzystywanych w działalności zawodowej.	P6U_W	P6S_WG
K_W05	w zaawansowanym stopniu rodzaje, techniki i narzędzia animacji komputerowej 2D/3D.	P6U_W	P6S_WG
K_W06	w zaawansowanym stopniu zagadnienia w zakresie programowania z wykorzystaniem języków C++, C#, JS, frameworków i biblioteki OpenGL	P6U_W	P6S_WG
K_W07	w zaawansowanym stopniu zagadnienia w zakresie architektury i sieci komputerowych, a także systemów operacyjnych, niezbędne do instalacji, konfiguracji oraz obsługi, utrzymania i zabezpieczania tych systemów.	P6U_W	P6S_WG
K_W08	w zaawansowanym stopniu zagadnienia w zakresie projektowania zjawisk fizycznych i mechaniki gier, w tym także związane z wykorzystaniem fotogrametrii.	P6U_W	P6S_WG
K_W09	w zaawansowanym stopniu zagadnienia związane z projektowaniem i obsługą systemów baz danych wykorzystywanych w działalności zawodowej związanej z kierunkiem studiów.	P6U_W	P6S_WG

K_W10	w zaawansowanym stopniu zagadnienia wchodzące w zakres inżynierii oprogramowania w tym dotyczących procesu implementacji, optymalizacji, wdrażania i testowania gier komputerowych, także dla środowisk sieciowych oraz VR/AR	P6U_W	P6S_WG
K_W11	w zaawansowanym stopniu zagadnienia związane z projektowaniem poziomów gier z uwzględnieniem fabuły, narracji, celu gry oraz funkcjonalności przestrzeni	P6U_W	P6S_WG
K_W12	w zaawansowanym stopniu zagadnienia w zakresie struktur danych i algorytmów przetwarzania informacji, analizy danych.	P6U_W	P6S_WG
K_W13	w zaawansowanym stopniu zagadnienia związane z procesem tworzenia gier na urządzenia mobilne oraz programowaniem sieciowym	P6U_W	P6S_WG
K_W14	w zawansowanym stopniu oraz wykorzystuje w praktyce nowoczesne narzędzia i techniki używane w produkcji gier, takie jak silniki Unity, Unreal Engine, a także różnorodne aplikacje graficzne, w tym narzędzia AI	P6U_W	P6S_WG
K_W15	w zaawansowanym stopniu zagadnienia w zakresie standardów i norm technicznych dotyczących informatyki; zna terminologię angielską/niemiecką z zakresu informatyki na poziomie B2 Europejskiego Systemu Opisu Kształcenia Językowego - oraz praktyczne zastosowanie tej wiedzy w działalności zawodowej związanej z kierunkiem studiów.	P6U_W	P6S_WG
K_W16	w zawansowanym stopniu zasady i technologie wykorzystywane w grafice 2D/3D, w tym zagadnienia związane z teorią koloru, kompozycją plastyczną projektowaniem i modelowaniem postaci, środowisk i obiektów, a także interfejsów użytkownika UI/UX	P6U_W	P6S_WG
K_W17	zagadnienia niezbędne do rozumienia pozatechnicznych uwarunkowań działalności inżynierskiej; podstawowe zasady bezpieczeństwa i higieny pracy obowiązujące w pracy programisty gier i grafika oraz wiedzę o kulturze fizycznej.	P6U_W	P6S_WK
K_W18	zagadnienia w zakresie ochrony własności przemysłowej, intelektualnej oraz prawa patentowego i autorskiego, a także zagadnienia z zakresu etyki zawodowej.	P6U_W	P6S_WK
K_W19	zagadnienia w zakresie zarządzania i prowadzenia działalności gospodarczej w tym specjalistycznych firm informatycznych szczególnie w branży gier komputerowych	P6U_W	P6S_WK
K_W20	zagadnienia w zakresie optymalizacji zasobów gry oraz zarządzania projektami w branży gier	P6U_W	P6S_WK
UMIEJĘTNOŚCI **Absolwent potrafi:**			
K_U01	pozyskiwać informacje z literatury, baz danych i innych źródeł; potrafi integrować uzyskane informacje, dokonywać ich interpretacji, a także wyciągać wnioski oraz formułować i uzasadniać opinie.	P6U_U	P6S_UU
K_U02	pracować indywidualnie i w zespole; umie oszacować czas potrzebny na realizację zleconego zadania; potrafi opracować i zrealizować harmonogram prac zapewniający dotrzymanie terminów oraz dba o zdrowie i sprawność fizyczną.	P6U_U	P6S_UO
K_U03	opracować dokumentację dotyczącą realizacji zadania inżynierskiego.	P6U_U	P6S_UW
K_U04	przygotować, przedstawić i omówić krótką prezentację poświęconą wynikom realizacji zadania inżynierskiego oraz brać udział w debacie.	P6U_U	P6S_UW; P6S_UK
K_U05	komunikować się w języku obcym na poziomie B2, w tym umożliwiającym czytanie dokumentacji technicznej wykorzystywanej w działalności zawodowej.	P6U_U	P6S_UK
K_U06	określić kierunki dalszego uczenia się i realizować proces samokształcenia.	P6U_U	P6S_UU
K_U07	wykorzystać poznane metody matematyczne, a także symulacje komputerowe do analizy, implementacji i oceny działania gier komputerowych	P6U_U	P6S_UW
K_U08	dokonać krytycznej analizy sposobów funkcjonowania systemów komputerowych i sieciowych oraz przeprowadzić diagnostykę tych systemów przy użyciu dostępnego oprogramowania i narzędzi sprzętowych.	P6U_U	P6S_UW
K_U09	porównać elementy systemów komputerowych i sieci ze względu na zadane kryteria użytkowe i ekonomiczne (bezpieczeństwo, niezawodność, szybkość działania, koszt itp.).	P6U_U	P6S_UW
K_U10	posłużyć się właściwie dobranymi środowiskami programistycznymi oraz narzędziami do projektowania, tworzenia, testowania i wdrażania gier komputerowych, w tym sieciowych i na urządzenia mobilne	P6U_U	P6S_UW
K_U11	tworzyć mechaniki gry oraz implementować rozgrywki w środowisku Unity i Unreal Engine	P6U_U	P6S_UW

K_U12	stosować algorytmy generowania proceduralnego oraz systemy symulacji fizyki w grach	P6U_U	P6S_UW
K_U13	zaprojektować poszczególne komponenty oprogramowania uwzględniające interaktywne interfejsy użytkownika przez prawidłowy dobór metod i narzędzi, w tym narzędzi AI	P6U_U	P6S_UW
K_U14	tworzyć koncepcje wizualne postaci, środowisk i obiektów przy użyciu stosownych narzędzi, w tym metod fotogrametrii.	P6U_U	P6S_UW
K_U15	projektować i edytować grafikę 2D, w tym tekstury, ikony oraz ilustracje promocyjne; tworzyć modele 3D, realistyczne i stylizowane tekstury dostosowane do stylu gry	P6U_U	P6S_UW
K_U16	zaplanować proces realizacji gry komputerowej, w tym jej wdrożenie; potrafi wstępnie oszacować koszty.	P6U_U	P6S_UW
K_U17	planować i tworzyć animacje postaci oraz efekty specjalne w tym uwzględniające oświetlenie i dźwięk	P6U_U	P6S_UW
K_U18	projektować poziomy gier uwzględniając fabułę, narrację, cel gry oraz funkcjonalność przestrzeni; tworzy immersyjne środowiska wirtualne i aplikacje VR/AR	P6U_U	P6S_UW
K_U19	sformułować algorytm procesu przetwarzania informacji, posługiwać się językami programowania wysokiego poziomu przez wykorzystanie odpowiednich narzędzi informatycznych.	P6U_U	P6S_UW
K_U20	dostrzegać przy formułowaniu i rozwiązywaniu zadań obejmujących projektowanie, wytwarzanie i wdrażanie gier komputerowych ich aspekty pozatechniczne, w tym środowiskowe, ekonomiczne i prawne.	P6U_U	P6S_UW
K_U21	podjąć pracę w środowisku przemysłowym, zwłaszcza w branży gier oraz potrafi bezpiecznie wykonywać zadania w swojej pracy przez stosowanie zasad BHP.	P6U_U	P6S_UW
K_U22	ocenić przydatność typowych metod i narzędzi służących do rozwiązywania informatycznych zadań inżynierskich oraz wybierać i stosować właściwe metody i narzędzia.	P6U_U	P6S_UW
K_U23	wykorzystać dostępne oprogramowanie do przetwarzania danych multimedialnych, w tym na potrzeby gier oraz reklamy i promocji w sieci.	P6U_U	P6S_UW
K_U24	projektować bazy danych; formułować zapytania do baz danych wykorzystując odpowiednie narzędzia.	P6U_U	P6S_UW
K_U25	korzystać ze standardów dotyczących projektowania, implementacji, testowania i użytkowania, obowiązujących w informatyce	P6U_U	P6S_UW
K_U26	testować i optymalizować gry pod kątem funkcjonalności, wydajności i UX	P6U_U	P6S_UW
KOMPETENCJE SPOŁECZNE Absolwent jest gotów:			
K_K01	ciągłego dokształcania się (studia drugiego i trzeciego stopnia, studia podyplomowe, kursy) — podnoszenia kompetencji zawodowych, osobistych i społecznych.	P6U_K	P6S_KK
K_K02	krytycznego respektowania pozatechnicznych aspektów i skutków działalności inżyniera, w tym ich wpływu na środowisko.	P6U_K	P6S_KK P6S_KO
K_K03	zachowywania się w sposób profesjonalny, przestrzegania zasad etyki zawodowej i poszanowania różnorodności poglądów i kultur, promuje społeczne i kulturowe znaczenie sportu.	P6U_K	P6S_KR
K_K04	przyjęcia odpowiedzialności za pracę własną oraz podporządkowania się zasadom pracy w zespole i ponoszenia odpowiedzialności za podejmowane decyzje i wspólnie realizowane zadania.	P6U_K	P6S_KO
K_K05	myślenia i działania w sposób przedsiębiorczy.	P6U_K	P6S_KO
K_K06	krytycznego formułowania i przekazywania społeczeństwu — m.in. poprzez środki masowego przekazu — informacji i opinii dotyczących osiągnięć informatyki, elektrotechniki i innych aspektów działalności inżyniera; jest gotów do przekazywania takich informacji i opinii w sposób powszechnie zrozumiały, w tym w języku obcym.	P6U_K	P6S_KK

3. WYMIAR, ZASADY I FORMA ODBYWANIA PRAKTYK ZAWODOWYCH, LICZBA ECTS DLA KIERUNKU STUDIÓW

Praktyki zawodowe realizowane są w wymiarze 6 miesięcy (38 punktów ECTS), a szczegółowe efekty uczenia się na praktykach zawodowych określa Program Praktyk Zawodowych i Dzienniczek praktyk zawodowych oraz sylabus dla kierunku Informatyka I stopień profil praktyczny.
Warunki zaliczania przez studentów WSEI efektów uczenia się na praktykach zawodowych określa Uchwała Senatu WSEI w Lublinie, zgodnie z którą praktyka zawodowa podzielona jest na dwie części:

I. Praktykę zawodową realizowaną na Uczelni,

II. Praktykę zawodową realizowaną u pracodawcy

Część pierwsza praktyki odbywa się wg następującego schematu:

- Wstęp do praktyk zawodowych – 25 godzin dydaktycznych na I semestrze studiów (1 ECTS)
- Projekt związany z kierunkiem studiów – 50 godzin dydaktycznych na IV semestrze studiów (2 ECTS)
- Projekt związany z kierunkiem studiów oraz raport z praktyki zawodowej – 75 godzin na VI semestrze studiów (3 ECTS)

Część druga praktyki zawodowej obejmuje 810 godzin dydaktycznych i odbywa się w terminie od 1 czerwca do 30 września danego roku odpowiednio w II, IV i VI semestrze po ukończeniu zajęć dydaktycznych. Student za realizację tej części otrzymuje 32 ECTS. Zatwierdzenie poszczególnych części praktyk zawodowych realizowanych u pracodawcy przez opiekuna praktyk zawodowych i przez dziekana następuje najpóźniej do 30 września każdego roku

4. WYBÓR MODUŁÓW ZAJĘĆ PRZEZ STUDENTÓW ZAWARTYCH W PROGRAMIE STUDIÓW

Liczba punktów ECTS, którą student uzyskuje realizując zajęcia podlegające wyborowi: **73** punkty ECTS, co stanowi **35** % ogólnej liczby punktów ECTS w programie. Do modułów do wyboru zostały zaliczone:

- język obcy (j. angielski, j. niemiecki) – 7 punktów ECTS,
- moduł fakultatywny: 4 punkty ECTS
- moduły specjalnościowe (9 modułów) – 56 punktów ECTS,
- seminarium i egzamin dyplomowy – 6 punktów ECTS,

5. LICZBA PUNKTÓW ECTS KSZTAŁTUJĄCA UMIEJĘTNOŚCI PRAKTYCZNE W PROGRAMIE STUDIÓW O PROFILU PRAKTYCZNYM

W programie studiów o profilu praktycznym na kierunku Informatyka określono liczbę punktów ECTS na 141, które kształtują umiejętności praktyczne, co stanowi 67% ogólnej liczby punktów ECTS w programie

6. ZAJĘCIA LUB GRUPY ZAJĘĆ WRAZ Z PRZYPISANIEM DO NICH EFEKTÓW UCZENIA SIĘ I TREŚCI PROGRAMOWYCH ZAPEWNIAJĄCYCH UZYSKANIE TYCH EFEKTÓW

Lp.	Moduł/ Przedmiot:	Efekty uczenia się w zakresie:			Treści programowe:	Forma zaliczenia (ZAO, EGZ)	Sposób weryfikacji i oceny efektów uczenia się
		wiedzy	umiejętności	kompetencji społecznych			
1.	Moduł ogólny	K_W18	K_U01, K_U21	K_K02 - K_K04	Wykłady: 1. Prezentacja systemów informatycznych Lubelskiej Akademii WSEI. Prezentacja platformy e-learningowej, wirtualnego dziekanatu, mobilnego studenta. 2. Bezpieczeństwo i higiena pracy – istota BHP, prawne aspekty BHP, prawo pracy, obowiązki pracodawców, prawa i obowiązki pracowników, Państwowa Inspekcja Pracy. 3. Bezpieczeństwo i higiena pracy – zagrożenia na stanowisku pracy i metody ich eliminowania lub ograniczania, ergonomia, wypadki przy pracy, choroby zawodowe, poszanowanie środowiska naturalnego i przestrzeganie zasad ekologii w pracy zawodowej. 4. Pojęcie i źródła prawa własności intelektualnej. Dobra niematerialne jako przedmiot ochrony. 5. Prawo autorskie i prawa pokrewne. Przedmiot ochrony prawa autorskiego - pojęcie utworu, rodzaje utworów (opracowania cudzych utworów, utwory inspirowane), przesłanki ochrony. 6. Korzystanie z materiałów źródłowych z poszanowaniem zasad prawa autorskiego. Sankcje z tytułu naruszania dóbr własności intelektualnej objętej ochroną prawną. 7. Własność intelektualna w branży IT. Licencje i rodzaje licencjonowania produktów IT. 8. Podstawowe zagadnienia ICT (domeny, certyfikaty ssl, podstawowe sieci komputerowe, systemy operacyjne) 9. Bezpieczeństwo informatyczne (zasady bezpiecznego korzystania z Internetu, zasady haseł, bezpieczne przetwarzanie danych osobowych, postepowanie w przypadku incydentów)	ZAO	Test, zadanie praktyczne

2a.	Język obcy (angielski)	K_W15	K_U05	K_K01	Ćwiczenia: Semestr III 1. Stosowanie czasów gramatycznych – powtórzenie i rozwinięcie wybranego materiału o nowe zastosowania, 2. Tworzenie pytań o podmiot i dopełnienie, 3. Stosowanie właściwego rejestru językowego w korespondencji: formalny i nieformalny list, email, 4. Posługiwanie się słownictwem z zakresu rodzina i relacje międzyludzkie – ćwiczenia leksykalne oraz typowe kolokacje, 5. Słabe i mocne formy czasowników posiłkowych, 6. Modele intonacji w formach pytających, 7. Jak pisać i mówić płynniej – łączniki zdań, 8. Zasady pisania krótkich artykułów prasowych, 9. Stosowanie formalnych i nieformalnych zwrotów grzecznościowych na przykładzie rozmowy telefonicznej, 10. Opisywanie stanowiska pracy – ćwiczenia leksykalne, 11. Stosowanie języka specjalistycznego - kierunkowego. Semestr IV 1. Stosowanie czasów gramatycznych – powtórzenie i rozwinięcie wybranego materiału o nowe zastosowania, 2. Jak powiedzieć to samo, ale inaczej - parafrazowanie wypowiedzi. 3. Wykorzystywanie różnorodnych form przymiotników w wypowiedziach na temat postaw i stereotypów – struktury porównawcze, regularne i nieregularne stopniowanie przymiotników, przymiotniki mocne. 4. Mówienie o przyzwyczajeniach i preferencjach - konstrukcje: used to, get used to, would rather and wish. 5. Wyrażanie umiejętności, zdolności i konieczności za pomocą czasowników modalnych. 6. Wyrażanie prośby za pomocą pytań pozornych. 7. Wypowiadanie się na temat podróżowania i środków transportu – słownictwo tematyczne. 8. Wykorzystywanie języka sugestii, sposoby komentowania i opiniowania. 9. Pisanie listu aplikacyjnego i cv – forma i zasady jego tworzenia,	EGZ	Testy semestralne prace domowe (ustne i pisemne), aktywność na zajęciach, Końcowy egzamin pisemny

					10. Rzeczowniki wieloznaczne na przykładzie cech osobowych. 11. Pisanie krótkich artykułów reklamowych. 12. Charakterystyka wypowiedzi pisemnych w esejach typu „za i przeciw” 13. Stosowanie języka specjalistycznego - kierunkowego Semestr V 1. Stosowanie czasów gramatycznych – powtórzenie i rozwinięcie wybranego materiału o nowe zastosowania, 2. Mówienie o sytuacjach hipotetycznych z wykorzystaniem okresów warunkowych, 3. Kozatywne użycie czasownika „have” i „get” 4. Hipotetyczność sytuacji – emocje i udzielanie rad – zadania leksykalno-gramatyczne, 5. Stosowanie czasowników frazalnych, 6. Przyimki i wyrażenia przyimkowe, 7. Wyrażanie ilości za pomocą kwantyfikatorów, 8. Strona bierna. 9. Formułowanie zdań złożonych podrzędnie i nadrzędnie, 10. Recenzja książki, filmu, ulubionej strony internetowej – zdania złożone i zasady tworzenia paragrafów, 11. Stosowanie języka specjalistycznego - kierunkowego.		
2b.	Język obcy (niemiecki)	K_W15	K_U05	K_K01	Ćwiczenia: Semestr III 1. Stosowanie czasów gramatycznych – powtórzenie i rozwinięcie wybranego materiału o nowe zastosowania, 2. Tworzenie pytań o podmiot i dopełnienie, 3. Stosowanie właściwego rejestru językowego w korespondencji: formalny i nieformalny list, email, 4. Posługiwanie się słownictwem z zakresu rodzina i relacje międzyludzkie – ćwiczenia leksykalne oraz typowe kolokacje, 5. Słabe i mocne formy czasowników posiłkowych, 6. Modele intonacji w formach pytających, 7. Jak pisać i mówić płynniej – łączniki zdań, 8. Zasady pisania krótkich artykułów prasowych, 9. Stosowanie formalnych i nieformalnych zwrotów grzecznościowych na przykładzie rozmowy telefonicznej,	EGZ	Testy semestralne prace domowe (ustne i pisemne), aktywność na zajęciach, Końcowy egzamin pisemny

					10. Opisywanie stanowiska pracy – ćwiczenia leksykalne, 11. Stosowanie języka specjalistycznego - kierunkowego. Semestr IV 1. Stosowanie czasów gramatycznych – powtórzenie i rozwinięcie wybranego materiału o nowe zastosowania, 2. Jak powiedzieć to samo, ale inaczej - parafrazowanie wypowiedzi. 3. Wykorzystywanie różnorodnych form przymiotników w wypowiedziach na temat postaw i stereotypów – struktury porównawcze, regularne i nieregularne stopniowanie przymiotników. 4. Mówienie o przyzwyczajeniach i preferencjach. 5. Wyrażanie umiejętności, zdolności i konieczności za pomocą czasowników modalnych. 6. Wyrażanie prośby za pomocą pytań pozornych. 7. Wypowiadanie się na temat podróżowania i środków transportu – słownictwo tematyczne. 8. Wykorzystywanie języka sugestii, sposoby komentowania i opiniowania. 9. Pisanie listu aplikacyjnego i cv – forma i zasady jego tworzenia, 10. Rzeczowniki wieloznaczne na przykładzie cech osobowych. 11. Pisanie krótkich artykułów reklamowych. 12. Charakterystyka wypowiedzi pisemnych w esejach typu „za i przeciw” 13. Stosowanie języka specjalistycznego - kierunkowego Semestr V 1. Stosowanie czasów gramatycznych – powtórzenie i rozwinięcie wybranego materiału o nowe zastosowania, 2. Mówienie o sytuacjach hipotetycznych z wykorzystaniem okresów warunkowych, 3. Hipotetyczność sytuacji – emocje i udzielanie rad – zadania leksykalno-gramatyczne, 4. Przyimki i wyrażenia przyimkowe, 5. Wyrażanie ilości za pomocą kwantyfikatorów, 6. Strona bierna.		

					7. Formułowanie zdań złożonych podrzędnie i nadrzędnie, 8. Recenzja książki, filmu, ulubionej strony internetowej – zdania złożone i zasady tworzenia paragrafów, 9. Stosowanie języka specjalistycznego - kierunkowego.		
3.	Moduł społeczno humanistyczny	K_W18	K_U01, K_U20	K_K03, K_K06	Wykłady: Psychologiczne aspekty funkcjonowania człowieka 1. Psychologia jako nauka i psychologiczne koncepcje człowieka • Dziedziny psychologii • Cele psychologii jako nauki (opis, wyjaśnianie, prognoza, kontrola) • Wybrane metody badań psychologicznych • Wspólne założenia psychologicznych koncepcji człowieka • Wybrane koncepcje (psychoanalityczna koncepcja człowieka behawiorystyczna koncepcja człowieka, koncepcja natury ludzkiej w psychologii humanistycznej, poznawcza koncepcja człowieka) 2. Percepcja i spostrzeganie społeczne • Pojęcie percepcji – jak odbieramy informacje • Spostrzeganie jako interpretacja wrażeń • Organizacja procesów spostrzegania (figura tło, bliskość, podobieństwo, domykanie) • Stałość procesów percepcyjnych a złudzenia • Wiedza i sądy o innych ludziach • Klasyczna teoria atrybucji f. Heidera • Deformacje procesu atrybucji • Zjawisko hallo-efektu i jego rodzaje 3. Podstawowe procesy poznawcze: pamięć, myślenie i rozwiązywanie problemów • Pojęcie i natura pamięci • Fazy procesu pamięciowego • Cechy i rodzaje pamięci • Blokowy model pamięci (pamięć sensoryczna, krótkotrwała, długotrwała) • Teorie dotyczące zapominania (zanik śladów pamięciowych interferencja, utrata dostępu, wypieranie) • Mnemotechniki • Definicje i natura procesu myślenia • Składniki procesu myślowego • Rola myślenia w rozwiązywaniu problemów • Irracjonalność w myśleniu i zniekształcenia poznawcze Filozoficzno- etyczne aspekty funkcjonowania człowieka	ZAO	Kolokwium zaliczeniowe, aktywność studenta na zajęciach, dyskusja podczas zajęć nad konkretnym Case study

					1. Podstawy filozofii jako umiłowania mądrości • Geneza, przedmiot i pojęcie filozofii • Podstawowe problemy filozoficzne i stanowiska filozoficzne w starożytności (Arche i zagadnienie zmiany oraz jego odniesienia epistemologiczne) • Sokrates: poglądy i narodziny Etyki jako nauki • Idealistyczna i racjonalistyczna koncepcja rzeczywistości i jej wpływ na późniejsze postrzeganie świata 2. Podstawy etyki • Przedsokratejskie kodeksy etyczne • Etyka jako nauka • Podstawowe pojęcia etyczne • Pojęcie czynu ludzkiego i czynniki wpływające na ocenę ludzkiego postępowania 3. Wybrane zagadnienia z etyki zawodowej • Etyka pracy: obowiązki wobec siebie i wobec pracodawcy • Podstawowe wartości etyki zawodowej: (prawda, kłamstwo, solidność, rzetelność, tajemnica) • Kodeksy etyczne: Kodeks etyczny PTI (Standardy etyczne)		
4.	Wychowanie fizyczne	K_W17	K_U02	K_K03	Ćwiczenia: 1. Omówienie zasad bezpieczeństwa i higieny w czasie zajęć wychowania fizycznego. Zapoznanie z materiałem nauczania i zasadami zaliczenia z zajęć. Zapoznanie z regulaminem i zasadami postępowania na sali gimnastycznej i siłowni. 2. Ćwiczenia ruchowe w utrzymaniu zdrowia człowieka. 3. Poznanie poszczególnych elementów związanych z piłką nożną, siatkową i koszykową – elementy techniki i taktyki, zasady sędziowania. 4. Omówienie form kształtowania sylwetki w ćwiczeniach aerobowych z wykorzystaniem różnorodnych przyborów tj. skakanek, taśm, hantli, piłek, body shaperów, ławeczek gimnastycznych. 5. Poznanie zasad stretchingu i callaneticsu jako skutecznej metody stopniowego rozciągania i kształtowania mięśni. 6. Zastosowanie poszczególnych form zajęć do kształtowania własnego ciała. 7. Znaczenie mikro i makroelementów w prawidłowym funkcjonowaniu organizmu podczas pracy tlenowej i wypoczynku. 8. Przedstawienie i zapoznanie z różnymi formami treningu stacyjnego i obwodowego.	ZAL	Zaliczenie na podstawie aktywnego udziału w dyskusji na temat kultury i sprawności fizycznej, zdrowia oraz higieny

					9. Omówienie aktywności fizycznej na poszczególnych etapach rozwoju człowieka, istota ruchu a proces starzenia. 10. Przedstawienie różnych form aktywności fizyczne z muzyką i bez: Fitness, Aerobik, Pilates, Body Shape, Aeroboxing - wpływ na kształtowanie poszczególnych grup mięśniowych. 11. Technika wykonywania ćwiczeń w różnych tempach, dobór właściwych obciążeń na poszczególne partie mięśniowe, omówienie czynnego wypoczynku i sprawności fizycznej do jakości życia codziennego i pracy zawodowej. 12. Ćwiczenia na mm. obręczy barkowej i biodrowej, mm. brzucha i pleców jako forma wzmacniająca mięśnie posturalne. 13. Omówienie istoty koncentracji i oddychania w ćwiczeniach fizycznych na układ sercowo - naczyniowy i oddechowy. 14. Piłka lekarska w zastosowaniu różnych form wzmacniających obręcz barkową, mm. brzucha i grzbietu, przysiady i półprzysiady z wyskokiem, przyjmowanie pozycji „WZ" i „S". 15. Podsumowanie, omówienie i ocena sprawności fizycznej.		
5.	Systemy operacyjne	K_W07			Wykłady: 1. System komputerowy a system operacyjny – definicje. 2. Podstawowe funkcjonalności systemu operacyjnego. 3. Systemy operacyjne w ujęciu historycznym. 4. Charakterystyka systemów rodziny MS Windows 5. Charakterystyka systemów klasy Linux i Android 6. Najważniejsze pliki i katalogi w system. 7. Podstawowe komendy konsoli. 8. Zarządzanie uprawnieniami i użytkownikami. 9. Zarządzanie procesami w systemie Linux i monitorowanie jego pracy	ZAO	Test pisemny
			K_U08	K_K03	Laboratoria:. 1. Podstawy administrowania systemem Windows: zarządzanie użytkownikami, struktura katalogów. 2. Administracja systemem Linux: zarządzanie kontami, struktura katalogów. 3. Polecenia zarządzania użytkownikami Windows, uprawnienia do plików i katalogów, listy ACL, polityki uprawnień. 4. Prawa dostępu w systemie Linux, tworzenie grup, uprawnienia szczegółowe - listy ACL 5. Rozruch systemu Windows – tryby rozruchu, odczyt konfiguracji, monitorowanie systemu i procesów. 6. Linux podstawowe operacje na plikach i katalogach: tworzenie, usuwanie, przenoszenie, kasowanie, zmiana nazw, montowanie systemów plików		Kolokwium pisemne, praktyczny test poleceń systemu Linux

					7. Zmienne środowiskowe Windows, definiowanie zmiennych, budowanie skryptów linii poleceń. 8. Zmienne środowiskowe Linux, definiowanie zmiennych, budowanie skryptów bash. 9. Dzienniki systemowe Windows, udostępnianie zasobów, harmonogramowanie zadań. 10. Rozruch systemu Linux – poziomy uruchomienia, odczyt konfiguracji, zarządzanie procesami w systemie i monitorowanie jego pracy. 11. Harmonogramy zadań w systemie Linux, definiowanie zadań.		
6.	Podstawy programowania	K_W06, K_W10			Wykłady: 1. Podstawowe informacje o programowaniu i językach programowania 2. Kompilacja i uruchamianie programów w środowisku Windows. 3. Język C++ – krótki przegląd. 4. Zmienne, stałe, operatory, wyrażenia, dyrektywy preprocesora. 5. Instrukcje strukturyzujące. 6. Tablice i łańcuchy znakowe. 7. Wskaźniki. 8. Funkcje. 9. Dostęp do danych i złożone typy danych. 10. Standardowe wejście-wyjście i operacje plikowe. 11. Złożone operatory, zarządzanie pamięcią, data i czas. 12. Preprocesor. 13. Standard C99.	EGZ	Egzamin pisemny
			K_U19	K_K02, K_K04	Laboratoria: 1. Instalacja i konfiguracja środowiska programisty. 2. Program wykonujący podstawowe obliczenia matematyczne, np. obliczający pola figur i objętości brył. 3. Rozwiązywanie równania kwadratowego w pełnym zakresie zmienności parametrów A,B,C. 4. Program obliczający odległości między miastami ze zbioru zapisanego w tablicy dwuwymiarowej. 5. Ciąg arytmetyczny, geometryczny, ciąg Fibonacciego. 6. Programowanie w drugiej dekadzie XXI wieku. 7. Inne problemy programistyczne ilustrujące podstawowe konstrukcje strukturalne języka C.		Aktywność na zajęciach i praca własna ewaluowana
7.	Architektura systemów komputerowych	K_W04, K_W07		K_K01,	Wykłady: 1. System komputerowy - wprowadzenie. Ewolucja komputerów – aspekt wydajnościowy. Komputery przyszłości. Klasyfikacja architektur komputerów. Funkcje komputera i jego podstawowa struktura	EGZ	Egzamin pisemny

					2. Systemy liczbowe. Arytmetyka liczb binarnych. Algebra Boole'a. Bramki logiczne. Układy kombinacyjne i sekwencyjne. Urządzenia logiczne o programowalnej strukturze - PLD 3. Budowa centralnej jednostki przetwarzania. Podstawowa zasada działania CPU – cykl rozkazowy, przetwarzanie potokowe. Listy rozkazów. Architektura CISC i RISC. Organizacja równoległa 4. Moduły I/O. Operacje wejścia-wyjścia. Magistrale komunikacyjne. Architektura płyty głównej. 5. Pamięć – podstawowe pojęcia. Hierarchia pamięci. Koncepcja pamięci Cache. Pamięć wewnętrzna – RAM, ROM. Pamięć masowa. Macierze dyskowe. 6. Podsystem graficzny i audio. Karta graficzna – architektura. GPU vs. CPU. Interfejsy karty graficznej. Digitalizacja dźwięku, budowa i funkcje karty dźwiękowej. Złącza audio 7. Obudowy komputerowe. Zasilacze impulsowe, UPS-y. Systemy chłodzenia. 8. Rodzaje transmisji danych. Interfejsy urządzeń peryferyjnych. Charakterystyka wybranych urządzeń wejścia-wyjścia. 9. Modyfikacje i diagnostyka komputera		
			K_U01, K_U03 K_U08, K_U09	K_K01, K_K06	Laboratoria: 1. Konwersja systemów liczbowych, działania arytmetyczne na liczbach binarnych 2. Podstawy projektowania i symulacji cyfrowych układów logicznych – układy kombinacyjne (sumator, komparator) i sekwencyjne (rejestr, licznik, ROM) 3. Diagnostyka komponentów komputera - BIOS, UEFI, POST - systemowe narzędzia diagnostyczne - pozasystemowe narzędzia diagnostyczne 4. Projekt i wykonanie prezentacji na temat związany z architekturą systemów komputerowych (do wyboru) 5. Projekt stanowiska komputerowego wg założonych kryteriów. Dobór komponentów systemu komputerowego wraz z uzasadnieniem		Prezentacja, sprawozdanie z laboratorium
8.	Wprowadzenie do grafiki komputerowej	K_W02 K_W16 K_W17		K_K01	Wykłady: 1. Podstawowe pojęcia związane z kompozycją i kolorem. 2. Grafika wektorowa. Rozdzielczość obrazu a ich wielkość rzeczywista i jakość wydruku. Zasady pracy i przykłady wykorzystania grafiki wektorowej. Narzędzia i interfejs programu Adobe Illustrator. Praca z krzywymi, kształtami,	EGZ	Egzamin w postaci testu

					tekstem. Narzędzie Pen tool. Filtry ścieżek. Różne rodzaje wypełnień, praca z gradientami. Paterny oraz ich wykorzystanie. 3. Zasady pracy z warstwami, obiektami, grupami obiektów w Adobe Illustrator. Korzystanie z wzorników kolorów. Przezroczystość, efekty, mieszanie warstw. Wykorzystanie narzędzie AI programu Adobe Illustrator. 4. Zasady pracy z projektami do druku oraz przygotowanie projektu do druku. 5. Grafika rastrowa. Zasady pracy i przykłady wykorzystania grafiki rastrowej. Narzędzia i interfejs programu Adobe Photoshop. 6. Praca na warstwach, maski i ich wykorzystanie. Warstwy dopasowania. Wykorzystanie obiektów typu Smart. Mieszanie warstw, style warstw. Wykorzystanie narzędzi AI programu. Adobe Photoshop. Efekty. Praca z kanałem Alpha. Eksport plików, najpopularniejsze formaty eksportu plików 7. Grafika 3d. Podstawowe zasady pracy z grafiką 3d do gier komputerowych. Interfejs i narzędzia programu Blender. Najważniejsze skróty klawiaturowe. Narzędzie Snap oraz Proportional editing. 8. Elementy składowe obiektów 3d - Wierzchołki (verticles), krawędzie (edges), ścianki(faces).Origin point (pivot). Rozmieszczanie obiektów względem innych obiektów. Grupowanie, łączenie, rozdzielanie obiektów. Podstawowe modyfikatory (Array, Simple deform, itp) Tryby wyświetlanie viewportu Blendera. Praca w trybie Object/Edit. 9. Tworzenie prostych scen z podstawowych brył geometrycznych, Manipulowanie bryłami. Dodawanie materiałów. 10. Oświetlenie. Rendering Cycles. Wykorzystanie Blender Kit. Oświetlenie mapa HDR.		
			K_U01 K_U02 K_U06 K_U15 K_U23 K_U18	K_K01 K_K04	Laboratoria: 1. Praca nad projektem z wykorzystaniem grafiki wektorowej w programie Adobe Illustrator. Zadanie polega na odwzorowaniu projektu w wektorach bazując na bitmapie, np. na podstawie starego plakatu filmowego. Używanie narzędzia pen tool, praca z filtrami ścieżek. Zarządzanie kolejnością obiektów na		3 zadania laboratoryjne - Projekt -grafika wektorowa. Projekt -grafika

					warstwach. Różne rodzaje wypełnienia oraz obrysu. Używanie panelu Wygląd (Apparance) Opcje narzędzia obrysu. 2. Praca nad projektem z wykorzystaniem grafiki rastrowej w programie Adobe photoshop. Przygotowanie kolażu fotograficznego z wykorzystaniem elementów wyciętych z wielu bitmap. Wykorzystanie narzędzi zaznaczania. Wtapianie - feather. Object select. Zaznaczanie metodą Pen Tool Praca na warstwach, maski i ich wykorzystanie. Warstwy dopasowania. Wykorzystanie obiektów typu Smart. Mieszanie warstw, style warstw. Wykorzystanie narzędzi AI programu Adobe Photoshop. Efekty. Praca z kanałem Alpha. Eksport plików, najpopularniejsze formaty eksportu plików. 3. Zbudowanie, oświetlenie i wyrenderowanie prostej sceny 3d: Zebranie referencji. Ustawienie najważniejszych elementów sceny z prostych brył geometrycznych z wykorzystaniem narzędzia Snap. Nauka grupowania obiektów - korzystanie z okna Outlinera oraz kolekcji. Materiały blendera. Przypisywanie do obiektów. Podpinanie tekstur. Korzystanie z gotowych bibliotek cyfrowych – BlenderKit. Oświetlenie sceny. Rodzaje świateł. kamera - ustawienia, opcje, efekty. Rendering w silniku Cycles.		rastrowa. Projekt - Render 3d
9.	Analiza matematyczna z algebrą liniową	K_W03		K_K01	Wykłady: 1. Pojęcie ciągu liczbowego i jego podstawowe własności, granica ciągu liczbowego oraz podstawowe metody wyznaczania granic ciągów, w tym ciągów Eulera. Funkcje wykładnicza i logarytmiczna o podstawie naturalnej. 2. Pojęcia granicy i ciągłości funkcji jednej zmiennej oraz własności takich funkcji. 3. Pojęcie pochodnej funkcji, jej własności i zastosowania, w tym twierdzenie de l'Hospitala. Pojęcie ekstremum funkcji jednej zmiennej, warunki konieczny i wystarczający istnienia ekstremum lokalnego. Wartości najmniejsza i największa funkcji na przedziale domkniętym. Ekstrema lokalne i globalne. Funkcje wypukłe i wklęsłe oraz ich zastosowania. 4. Pojęcie funkcji pierwotnej oraz metody wyznaczania funkcji pierwotnych. Całka oznaczona i metody jej wyznaczania oraz zastosowania całki oznaczonej.	ZAO	Test pisemny

					5. Algebra macierzy. Wyznacznik macierzy kwadratowej. Własności wyznaczników. Macierz odwrotna. Zastosowanie macierzy i wyznaczników w informatyce.		
			K_U07	K_K01	Ćwiczenia: 1. Przegląd funkcji elementarnych w tym ich własności i wykresów. 2. Obliczanie granic ciągów i funkcji oraz wyznaczanie pochodnych funkcji. 3. Wykorzystanie rachunku pochodnych do wyznaczania granic funkcji, ekstremów lokalnych i ekstremów globalnych, badania wklęsłości i wypukłości funkcji. 4. Wyznaczanie całek nieoznaczonych i oznaczonych funkcji jednej zmiennej. Zastosowanie rachunku całkowego. 5. Wykonywanie działań na macierzach, obliczanie wyznacznika macierzy orz macierzy odwrotnej.		Kolokwium pisemne (6 zadań). Aktywność na zajęciach
10.	Algorytmy i struktury danych	K_W01, K_W12		K_K01	Wykłady: 1. Komputerowa reprezentacja liczb zmiennopozycyjnych. Błędy zaokrągleń w procesach numerycznych. 2. Interpolacja numeryczna (wzór Lagrange'a oraz wzór Newtona). 3. Całkowanie numeryczne (kwadratury interpolacyjne). 4. Rozwiązywanie układów równań liniowych i eliminacja Gaussa. 5. Numeryczne rozwiązywanie równań nieliniowych (metody Newtona i siecznych). 6. Algorytmy i ich złożoność. Listy liniowe (jedno i dwukierunkowe). Kolejki i stosy oraz ich zastosowania. 7. Struktury drzewiaste, binarne drzewa poszukiwań, drzewa czerwono-czarne. Zastosowanie B-drzew do tworzenia indeksów. 8. Kopce i ich zastosowanie do sortowania. Algorytm sortowania szybkiego. Sortowanie przez łączenie. 9. Grafy oraz podstawowe algorytmy grafowe	ZAO	Test
			K_U19	K_K03, K_K06,	Laboratoria: 1. Praktyczne rozwiązywanie problemów numerycznych przy użyciu programu GNU Octave. 2. Implementacja wybranych algorytmów i struktur danych w języku GNU OCTAVE 3. Zastosowanie algorytmów i struktur danych do rozwiązywania problemów praktycznych. 4. Zastosowanie metod numerycznych w rozwiązywaniu problemów technicznych. 5. Praktyczne wykorzystanie algorytmów przetwarzania nienumerycznego. Zakłada się, że przedstawiciel otoczenia gospodarczego przedstawi w formie prezentacji oraz analizy		Oceny za prace w grupach w ramach zadania laboratoryjnego. Obecność

					przypadków wybrane algorytmy oraz ich praktyczne zastosowanie		
11.	Programowanie w C++/C#	K_W06, K_W10			Wykłady: 1. Podstawy języka C++/C# – funkcje, instrukcje warunkowe, pętle, zakresy, stałe. Wprowadzenie do składni języka, obejmujące definiowanie funkcji, instrukcje warunkowe, pętle, zakresy zmiennych oraz deklarowanie stałych. 2. Typy danych i operacje – typy wbudowane, aliasy, auto, operacje arytmetyczne, inkrementacje. Omówienie typów wbudowanych, aliasów typów, słowa kluczowego auto oraz operacji arytmetycznych, ze szczególnym uwzględnieniem inkrementacji pre- i postfiksowej. 3. Tablice, wskaźniki i referencje – wskaźniki, referencje, stałe wskaźniki, różnice, niebezpieczeństwa. Przedstawienie podstawowych mechanizmów operowania na pamięci: deklaracji i użytkowania tablic, wskaźników oraz referencji, a także różnic między nimi. 4. Zaawansowane typy danych – enum, smart pointery, układ w pamięci. Analiza struktur danych takich jak wyliczenia (enum), inteligentne wskaźniki (smart pointers) oraz układ danych w pamięci komputera. 5. Standard Template Library (STL) – std::string, std::vector, std::list, std::map, pętle po kolekcji. Zastosowanie podstawowych struktur danych z biblioteki STL, takich jak std::string, std::vector, std::list i std::map, wraz z podstawami iteracji po kolekcjach. 6. Kontenery i iteratory – przegląd kontenerów STL, iteratory, porównanie std::vector vs std::list. Szczegółowe omówienie kontenerów STL oraz mechanizmów iterowania, ze wskazaniem różnic pomiędzy różnymi strukturami danych i ich efektywnością. 7. Algorytmy i złożoność obliczeniowa – algorytmy STL, podstawy złożoności. Zastosowanie wybranych algorytmów z biblioteki STL oraz wprowadzenie do analizy ich złożoności czasowej i pamięciowej. 8. Strumienie wejścia/wyjścia – operacje na strumieniach w C++. Podstawy obsługi strumieni w C++, w tym operacje wejścia i wyjścia za pomocą std::cin, std::cout oraz innych mechanizmów standardowych. 9. Programowanie obiektowe – klasy, obiekty, struktury, konstruktory, destruktory. Tworzenie klas i obiektów,	EGZ	Egzamin pisemny.

					definiowanie konstruktorów, destruktorów, metod dostępowych (settery i gettery) oraz podstawowe zasady enkapsulacji. 10. Dziedziczenie i polimorfizm – interfejsy, klasy abstrakcyjne, funkcje wirtualne, virtual/override. Wprowadzenie do dziedziczenia klas, funkcji wirtualnych, klas abstrakcyjnych oraz słów kluczowych virtual, override i static w kontekście programowania obiektowego. 11. Funkcje lambda i obiekty funkcyjne – nowoczesne podejście do przekazywania zachowań w kodzie. Praktyczne zastosowanie funkcji anonimowych (lambda) oraz obiektów funkcyjnych w celu zwiększenia elastyczności kodu i uproszczenia konstrukcji funkcyjnych. 12. Nowości w C++20 i C++23 – biblioteka ranges oraz widoki. Przegląd zmian w najnowszych wersjach standardu języka C++, obejmujących nowe składnie, rozszerzenia biblioteczne oraz usprawnienia kompilatora.		
			K_U10, K_U19		Laboratoria: 1. Pisanie prostych programów w C++ • Tworzenie programu z funkcją main, instrukcjami warunkowymi i pętlami. • Zadania: kalkulator, program sprawdzający liczby pierwsze, tabliczka mnożenia. 2. Praca z typami danych i konwersją • Ćwiczenia z auto, typedef, rzutowaniem, inkrementacją. • Zadania: konwerter jednostek, eksploracja typów auto i zachowania różnych operatorów. 3. Manipulacja pamięcią i adresami • Praktyczne zadania z wskaźnikami, referencjami, tablicami. • Zadania: implementacja własnej wersji funkcji strlen, zamiana wartości wskaźnikiem oraz referencją. 4. Wskaźniki inteligentne i wyliczenia w akcji • Tworzenie prostych klas z użyciem std::unique_ptr, enum class. • Zadania: symulacja zarządzania zasobami (np. menedżer zasobów). 5. Operacje na kontenerach STL • Praca z std::vector, std::string, std::map, iteracje po kolekcjach. • Zadania: licznik znaków, licznik słów w tekście, konwersja liczbowo-słowna. 6. Wydajność i wybór kontenera • Porównanie działania vector i list, użycie iteratorów.		Zadania laboratoryjne.

					• Zadania: implementacja sortowania, analiza czasu działania, pomiar zużycia pamięci. 7. Algorytmy STL w praktyce • Użycie std::sort, std::find, std::accumulate, std::transform. • Zadania: analiza danych (średnie, mediany), przekształcenia ciągów znaków. 8. Wejście/wyjście – pliki i użytkownik • Obsługa plików, odczyt/zapis, przetwarzanie danych z wejścia. • Zadania: prosty parser CSV, kopiowanie pliku, logowanie operacji. 9. Projektowanie klas i enkapsulacja • Tworzenie własnych klas, metod, konstruktorów i destruktorów. • Zadania: modelowanie obiektów (np. konto bankowe, książka, samochód). 10. Polimorfizm i hierarchie klas • Ćwiczenia z dziedziczeniem, funkcjami wirtualnymi, interfejsami. • Zadania: system zwierząt/pojazdów, obsługa wielu formatów plików. 11. Funkcje lambda i funkcje wyższego rzędu • Tworzenie lambd, sortowanie i filtrowanie za pomocą lambd. • Zadania: filtr listy, niestandardowe sortowanie, własny for_each. 12. Praca z ranges i nowościami z C++20/23 • Praktyczne użycie std::ranges, views::filter, views::transform. • Zadania: filtrowanie danych z pliku, transformacja danych, pipeline przetwarzania.		
12.	Wprowadzenie do animacji komputerowej	K_W02, K_W05, K_W14		K_K01	Wykłady: 1. Historia i ewolucja animacji: • Od tradycyjnej animacji rysunkowej po współczesne techniki cyfrowe. • Kluczowe postacie i dzieła w historii animacji. • Rola animacji w grach komputerowych, filmie i reklamie. 2. Podstawowe zasady animacji (12 zasad Disneya): • Zgniatanie i rozciąganie (Squash and Stretch). • Wyprzedzenie (Anticipation).	EGZ	Egzamin pisemny

					• Inscenizacja (Staging). • Rysowanie progresywne i od klatki kluczowej do klatki kluczowej (Straight Ahead Action and Pose to Pose). • Przenikanie i zachodzenie na siebie akcji (Follow Through and Overlapping Action). • Rozpędzanie i zwalnianie (Slow In and Slow Out). • Łuki (Arcs). • Akcja drugoplanowa (Secondary Action). • Taktowanie (Timing). • Wyolbrzymienie (Exaggeration). • Solidne rysowanie (Solid Drawing - w kontekście bryły i przestrzeni). • Urok (Appeal). 3. Klatki kluczowe (Keyframes) i interpolacja: • Definicja klatki kluczowej. • Rodzaje interpolacji (liniowa, krzywe Beziera, ease-in/ease-out). • Edytory krzywych animacji (Graph Editor/Curve Editor) i ich znaczenie. 4. Rodzaje animacji komputerowej: • Animacja 2D: tradycyjna cyfrowa, wektorowa, cut-out, sprite sheets. • Animacja 3D: modelowanie, rigging, skinning, animacja postaci, animacja obiektów. • Animacja proceduralna (wprowadzenie). • Motion Capture (MoCap) – zasady działania i zastosowania. 5. Podstawy animacji 2D: • Praca z warstwami i osiami czasu w oprogramowaniu 2D. • Tworzenie prostych cykli animacyjnych (np. chód, bieg dla postaci 2D). • Animacja klatek pośrednich (tweening). • Wykorzystanie arkuszy sprite'ów (sprite sheets) w grach. 6. Wprowadzenie do animacji 3D: • Proces tworzenia animacji 3D: od modelu do gotowej animacji. • Manipulacja obiektami w przestrzeni 3D: transformacje (przesunięcie, rotacja, skalowanie) w czasie. 7. Rigging postaci 3D (wprowadzenie): • Koncepcja szkieletu (armature/skeleton) i kości (bones). • Hierarchia kości i stawów (joints).		

					• Podstawy tworzenia prostego rigu dla postaci humanoidalnej. • Kontrolery (controllers) ułatwiające animację. 8. Skinning (Vertex Weighting) (wprowadzenie): • Proces przypisywania wierzchołków modelu do kości szkieletu. • Wpływ wag na deformację siatki podczas ruchu. • Podstawowe narzędzia do skinningu. 9. Animacja postaci 3D: • Tworzenie cykli animacyjnych (np. chód, bieg, skok, idle). • Wykorzystanie referencji wideo. • Animacja ekspresji twarzy (wprowadzenie do blendshapes/morph targets). 10. Animacja obiektów i efektów specjalnych: • Animowanie prostych mechanizmów, pojazdów. • Podstawy animacji efektów (np. eksplozje, particle). 11. Oprogramowanie do animacji: • Przegląd popularnych narzędzi do animacji 2D (np. Adobe Animate, Toon Boom Harmony, Spine, Krita). • Przegląd popularnych narzędzi do animacji 3D (np. Blender, Autodesk Maya, 3ds Max). • Integracja animacji z silnikami gier (np. Unity, Unreal Engine) – formaty plików, systemy animacji w silnikach. 12. Zastosowanie animacji w grach komputerowych: • Animacje gracza i postaci niezależnych (NPC). • Animacje interfejsu użytkownika (UI). • Cutscenki i animacje fabularne. • Animacje środowiskowe.		
			K_U13, K_U15, K_U17, K_U23	K_K01	Laboratoria: 1. Zapoznanie z interfejsem oprogramowania do animacji (np. Blender dla 3D, Krita/OpenToonz dla 2D): • Nawigacja, podstawowe narzędzia, oś czasu, edytor krzywych. 2. Ćwiczenia z zasad animacji – proste obiekty: • Animacja odbijającej się piłki (timing, squash & stretch, arcs). • Animacja wahadła (overlapping action, ease-in/ease-out). 3. Animacja klatkowa 2D: • Tworzenie prostego cyklu animacji 2D (np. mruganie okiem, prosty ruch ręką).		Zadania laboratoryjne

					4. Podstawy animacji 3D – transformacje obiektów: • Animowanie ruchu, rotacji i skalowania prostych brył 3D. • Praca z klatkami kluczowymi i edytorem krzywych. 5. Tworzenie prostego rigu 2D (cut-out): • Przygotowanie grafiki postaci 2D (podział na części). • Stworzenie prostego szkieletu i przypisanie części ciała. • Animacja prostej postaci 2D. 6. Podstawy riggingu postaci 3D: • Tworzenie prostego szkieletu dla uproszczonej postaci humanoidalnej lub stworzenia. • Ustawianie hierarchii kości. 7. Podstawy skinningu postaci 3D: • Automatyczne i ręczne przypisywanie wag wierzchołków do kości. • Testowanie deformacji. 8. Animacja cyklu chodu dla postaci 3D: • Analiza referencji. • Tworzenie kluczowych póz i interpolacja. • Dopracowywanie ruchu w edytorze krzywych. 9. Animacja prostego obiektu mechanicznego: • Animowanie np. otwierających się drzwi, prostego mechanizmu. 10. Eksport i import animacji do silnika gry (np. Unity/Unreal): • Przygotowanie animacji do eksportu (np. format FBX). • Import i konfiguracja animacji w silniku gry. • Tworzenie prostego kontrolera animacji (Animator Controller/Animation Blueprint – wprowadzenie). 11. Mini-projekt animacyjny: • Stworzenie krótkiej animacji postaci (2D lub 3D) lub obiektu, demonstrującej poznane techniki i zasady.		
13.	Zarządzanie projektami IT	K_W20			Wykłady: 1. Ogólne pojęcie projektu i jego cechy charakterystyczne, 2. Charakterystyka wybranych rodzajów projektów w tym schemat funkcjonowania faz projektów, 3. Cechy projektów informatycznych, 4. Obszary tematyczne projektów informatycznych a cechy kierownika projektu, 5. Wiedza i umiejętności kierownika projektu, 6. Zasady pracy zespołowej przy realizacji projektu, 7. Wybrane zagadnienia zasad negocjacji z interesariuszami wewnętrznymi i zewnętrznymi, przy realizacji projektu,	ZAO	Test pisemny

					8. Podstawy zarządzania projektami wdrożeniowymi. 9. Struktura prac wdrożeniowych. 10. Metodyki pracy z projektami IT 11. Zapytanie, oferty, specyfikacje funkcjonalne 12. Analiza przedwdrożeniowa 13. Przygotowanie dokumentu analizy przedwdrożeniowej 14. Raporty z konfiguracji, wdrożeń i szkoleń 15. Uruchomienie systemu. 16. Asysta powdrożeniowa		
			K_U16, K_U20, K_U22	K_K01, K_K04	Ćwiczenia: 1. Zapytanie ofertowe 2. Specyfikacja funkcjonalna 3. Ofertowanie 4. Harmonogram projektu 5. Pytania analityczne 6. Analiza konfiguracyjna 7. Dokumentacja projektowa		Ocena zadań
14.	Wprowadzenie do sieci komputerowych	K_W04, K_W07		K_K01	Wykłady: 1. Odkrywanie sieci. 2. Konfigurowanie sieciowego systemu operacyjnego. 3. Protokoły sieciowe i komunikacja. 4. Warstwa dostępu do sieci. 5. Ethernet. 6. Warstwa sieciowa. 7. Warstwa transportu. 8. Adresowanie IP. 9. Podsieci w sieciach IP. 10. Warstwa aplikacji. 11. Sieć komputerowa.	ZAO	Test pisemny
			K_U08, K_U09	K_K01, K_K04	Laboratoria: 1. Inicjowanie i przeładowywanie routera i przełącznika. Instalowanie konfiguracja protokołu IPv6 w systemie 10/11. Badanie narzędzi współpracy sieciowej. Badanie kowergentnych usług sieciowych. Odwzorowywanie Internetu. 2. Ustanawianie sesji konsoli za pomocą programu Tera Term. Budowanie prostej sieci. Konfigurowanie adresu zarządzania przełącznikiem. 3. Badanie Standardów sieciowych. 4. Obserwacja ruchu w sieci. 5. Obserwacja trzy etapowego nawiązywania połączenia TCP. 6. Wykorzystując program Wireshark do zbadania i przechwytywania ramek UDP protokołu DNS.		Oceny za pracę w ramach poszczególnych zadań laboratoryjnych

					7. Wykorzystanie programu Wireshark do zbadania i przechwytywania ramek protokołu FTP i HTTP. 8. Konwersja adresów IPv4 na binarne. Rozpoznawanie adresów IPv4. Rozpoznawanie adresów IPv6. Konfiguracja adresów IPv6 na urządzeniach sieciowych. 9. Projektowanie i wdrażanie schematu adresowania opartego na podsieciach IPv4. 10. Projektowanie i wdrażanie schematów adresowania VLSM. 11. Śledzenie adresów MAC urządzeń sieciowych. 12. Wykorzystanie programu Wireshark, do badania ramek Ethernet. Obserwacja ARP przy pomocy Windows CLI, IOS CLI oraz programu Wireshark. 13. Korzystanie z IOS CLI do badania tabel adresów MAC przełącznika		
15.	Produkcja gier komputerowych i aplikacji interaktywnych	K_W02, K_W18, K_W19, K_W20		K_K01	Wykłady: 1. Wprowadzenie do produkcji gier i aplikacji interaktywnych: • Definicja produkcji, rola producenta. • Specyfika branży gier komputerowych i aplikacji interaktywnych. • Typy studiów deweloperskich (Indie, AA, AAA) i ich modele produkcyjne. • Cykl życia produktu (gry/aplikacji). 2. Fazy produkcji gier: • Koncepcja (Concept): Generowanie pomysłów, analiza rynku, grupa docelowa, tworzenie dokumentu koncepcyjnego (Concept Document/Pitch Document). • Preprodukcja (Pre-production): Tworzenie GDD (Game Design Document), prototypowanie kluczowych mechanik, planowanie, budżetowanie, formowanie zespołu, wybór technologii. • Produkcja (Production): Właściwe tworzenie gry – grafika, programowanie, dźwięk, projektowanie poziomów. Iteracyjny rozwój, kamienie milowe (milestones). • Testowanie (Alpha, Beta): Wewnętrzne i zewnętrzne testy, QA (Quality Assurance), bug tracking. • Wydanie (Release/Launch): Marketing, dystrybucja (platformy cyfrowe, fizyczne), przygotowanie do premiery. • Postprodukcja (Post-launch): Wsparcie po premierze, aktualizacje, patche, DLC, analiza metryk, budowanie społeczności. 3. Role w zespole produkcyjnym: • Producent, Project Manager.	EGZ	Egzamin pisemny

					• Game Designer, Level Designer, Narrative Designer. • Programiści (Gameplay, Engine, Tools, AI, UI). • Graficy (2D Artist, 3D Artist, Concept Artist, Animator, UI Artist, Technical Artist). • Sound Designer, Kompozytor. • Testerzy QA. • Marketing & PR Specialist, Community Manager. • Role wspierające (HR, administracja). 4. Dokumentacja projektowa: • Game Design Document (GDD) – struktura i kluczowe elementy. • Technical Design Document (TDD). • Art Bible/Style Guide. • Plany produkcyjne, harmonogramy. 5. Metodyki zarządzania projektami w gamedevie: • Podejście kaskadowe (Waterfall) – wady i zalety. • Metodyki zwinne (Agile): Scrum (role, artefakty, ceremonie), Kanban. • Podejścia hybrydowe. • Narzędzia do zarządzania projektami (np. Trello, GitLab, Redmine). 6. Planowanie i budżetowanie projektu: • Szacowanie czasu i zasobów. • Tworzenie harmonogramu, identyfikacja ścieżki krytycznej. • Elementy budżetu (koszty osobowe, licencje, marketing, outsourcing). • Źródła finansowania projektów (wydawcy, inwestorzy, crowdfunding, granty). • Zarządzanie ryzykiem w projekcie. 7. Zarządzanie zespołem i komunikacja: • Budowanie i motywowanie zespołu. • Efektywna komunikacja wewnętrzna i zewnętrzna. • Rozwiązywanie konfliktów. • Kultura pracy w gamedevie, crunch time – aspekty etyczne. 8. Jakość i testowanie (Quality Assurance - QA): • Rola QA w procesie produkcji. • Rodzaje testów (funkcjonalne, wydajnościowe, kompatybilności, użyteczności, lokalizacyjne). • Zarządzanie błędami (bug tracking systems). • Cykle testowe (Alpha, Beta, Gold Master). • Automatyzacja testów (wprowadzenie).		

					9. Aspekty prawne i biznesowe: • Prawa autorskie i własność intelektualna w grach. • Licencjonowanie oprogramowania, silników, zasobów. • Umowy z wydawcami, deweloperami, freelancerami. • Modele biznesowe gier (Premium, Free-to-Play, subskrypcje, reklamy). • Dystrybucja cyfrowa (Steam, GOG, App Store, Google Play) i fizyczna. 10. Narzędzia i technologie wspierające produkcję: • Systemy kontroli wersji (np. Git, Perforce). • Platformy do współpracy i komunikacji (np. Slack, Discord, Microsoft Teams). • Narzędzia do tworzenia dokumentacji (np. Confluence, Google Docs). • Silniki gier jako platformy produkcyjne. 11. Outsourcing w produkcji gier: • Kiedy i co warto outsourcować (grafika, dźwięk, kodowanie, QA). • Wybór i zarządzanie dostawcami zewnętrznymi. • Wady i zalety outsourcingu. 12. Analiza post-mortem projektu: • Cel i znaczenie analizy po zakończeniu projektu. • Identyfikacja sukcesów, porażek i wniosków na przyszłość (lessons learned). • Metody przeprowadzania post-mortem.		
			K_U03, K_U04, K_U16, K_U20	K_K01, K_K02, K_K04, K_K05	Laboratoria: 1. Analiza studiów przypadku udanych i nieudanych produkcji gier: • Identyfikacja kluczowych czynników sukcesu/porażki. • Omówienie wniosków. 2. Tworzenie dokumentu koncepcyjnego (Pitch Document) dla własnego pomysłu na grę/aplikację: • Definiowanie głównej idei, grupy docelowej, USP (Unique Selling Proposition). 3. Podstawy tworzenia Game Design Document (GDD): • Opracowanie wybranego fragmentu GDD dla projektu (np. opis mechaniki, projektu postaci, zarysu fabuły). 4. Symulacja procesu planowania projektu (np. w Scrum): • Tworzenie Product Backlogu, szacowanie zadań (Story Points). • Planowanie Sprintu, tworzenie Sprint Backlogu.		Ocena za samodzielną pracę w ramach zadania laboratoryjnego

					• Prowadzenie Daily Scrum (symulacja). 5. Praca z narzędziami do zarządzania projektami: • Zakładanie projektu w wybranym narzędziu (np. Trello). • Tworzenie zadań, przypisywanie odpowiedzialności, śledzenie postępów. 6. Przygotowanie prostego harmonogramu i budżetu dla małego projektu: • Identyfikacja głównych zadań, szacowanie czasu. • Określenie podstawowych kategorii kosztów. 7. Zarządzanie ryzykiem – identyfikacja i planowanie reakcji: • Burza mózgów na temat potencjalnych ryzyk w przykładowym projekcie. • Propozycje działań mitygujących. 8. Symulacja procesu testowania i raportowania błędów: • Przygotowanie scenariuszy testowych dla prostej aplikacji/gry. • Raportowanie znalezionych błędów w systemie bug tracking (np. uproszczony arkusz). 9. Przygotowanie prezentacji marketingowej/pitchu dla projektu gry: • Skupienie się na kluczowych cechach i grupie docelowej. 10. Projekt grupowy – symulacja procesu preprodukcji/produkcji małej gry/aplikacji: • Podział ról w zespole. • Stworzenie kluczowej dokumentacji (uproszczone GDD, plan). • Prezentacja postępów i finalnego produktu (koncepcji/prototypu). • Przeprowadzenie uproszczonego post-mortem.		
16.	Projektowanie interaktywnych interfejsów użytkownika	K_W02, K_W10, K_W14, K_W16		K_K01	Wykłady: 1. Wprowadzenie do UI/UX: • Definicje UI (User Interface) i UX (User Experience), kluczowe różnice i powiązania. • Znaczenie projektowania zorientowanego na użytkownika (User-Centered Design) w tworzeniu gier i aplikacji interaktywnych. • Proces projektowania UX: etapy, iteracyjność. • Rola projektanta UI/UX w zespole deweloperskim. 2. Badania użytkowników (User Research):	EGZ	Egzamin pisemny

					• Metody badawcze: wywiady, ankiety, obserwacje, analiza danych. • Tworzenie person użytkowników, scenariuszy użycia (use cases) i map empatii. • Analiza konkurencji i identyfikacja dobrych praktyk. 3. Architektura Informacji (IA) i Nawigacja: • Podstawy IA: organizacja, strukturyzacja i etykietowanie treści. • Projektowanie przepływów użytkownika (user flows) i map stron/ekranów. • Rodzaje nawigacji w grach i aplikacjach (menu, HUD, interfejsy kontekstowe). 4. Podstawy projektowania interakcji (Interaction Design - IxD): • Zasady projektowania interakcji (np. prawa Fittsa, Hicka). • Wzorce interakcji i ich zastosowanie w grach. • Feedback dla użytkownika (wizualny, dźwiękowy, haptyczny). • Projektowanie afordancji i sygnifikatorów. 5. Wireframing i Prototypowanie: • Rodzaje wireframe'ów (low-fidelity, mid-fidelity, high-fidelity). • Narzędzia do tworzenia wireframe'ów i prototypów (np. Figma, Adobe XD, Balsamiq). • Tworzenie interaktywnych prototypów, testowanie koncepcji. 6. Zasady projektowania wizualnego (UI Design): • Teoria koloru, psychologia koloru w grach. • Typografia w interfejsach: czytelność, hierarchia, dobór fontów. • Kompozycja, layout, siatki (grids). • Ikonografia i jej rola w komunikacji wizualnej. • Tworzenie spójnego stylu wizualnego (UI Kit, Design System). 7. Specyfika UI/UX dla gier komputerowych: • Projektowanie HUD (Heads-Up Display), menu (główne, pauzy, opcji). • Interfejsy ekwipunku, map, drzewek umiejętności, systemów craftingu. • Projektowanie tutoriali i onboardingu gracza. • Game feel i jego związek z UI/UX. 8. UI/UX dla technologii immersyjnych (VR/AR):		

					• Wyzwania i specyfika projektowania interfejsów dla VR i AR. • Metody interakcji w środowiskach immersyjnych (kontrolery, gesty, wzrok). • Zapobieganie chorobie symulatorowej (motion sickness) poprzez design. 9. Testowanie użyteczności (Usability Testing): • Metody testowania użyteczności (testy korytarzowe, testy A/B, heurystyki Nielsena, eye-tracking). • Planowanie i przeprowadzanie sesji testowych. • Analiza wyników, raportowanie i iteracja projektu. 10. Dostępność (Accessibility) w projektowaniu UI/UX: • Projektowanie dla graczy i użytkowników z różnymi potrzebami (np. wzrokowymi, słuchowymi, motorycznymi). • Standardy dostępności (np. WCAG) i ich adaptacja w grach. • Narzędzia i techniki wspierające dostępność. 11. Narzędzia i technologie wspierające projektowanie UI/UX: • Przegląd popularnych narzędzi (np. Figma, Adobe XD, Sketch, Principle). • Integracja z silnikami gier (np. Unity UI, Unreal Engine UMG). • Wykorzystanie narzędzi AI we wspomaganiu projektowania. 12. Trendy i przyszłość UI/UX w grach i technologiach interaktywnych: • Adaptacyjne interfejsy, personalizacja. • Interfejsy głosowe (VUI), interfejsy mózg-komputer (BCI) – perspektywy. • Etyczne aspekty projektowania UI/UX.		
			K_U01, K_U03, K_U04, K_U13, K_U26	K_K01, K_K03, K_K04	Laboratoria: 1. Analiza UI/UX istniejących gier i aplikacji: • Dezintegracja interfejsów popularnych gier (różne gatunki i platformy). • Identyfikacja dobrych i złych praktyk, ocena pod kątem heurystyk użyteczności. 2. Tworzenie person i scenariuszy użytkownika: • Praktyczne ćwiczenia z definiowania grup docelowych i tworzenia person. • Opracowywanie user stories i user flows dla wybranego projektu gry/aplikacji.		Sprawozdanie z laboratorium – dokumentacja projektowa

					3. Projektowanie architektury informacji: • Tworzenie mapy serwisu/aplikacji (sitemap) i diagramów przepływu dla zadań projektowych. 4. Szkicowanie i wireframing: • Praktyczne tworzenie szkiców (paper prototyping) i cyfrowych wireframe'ów (low/mid-fidelity) dla kluczowych ekranów interfejsu. • Wykorzystanie wybranego narzędzia do wireframingu. 5. Prototypowanie interaktywne: • Budowa klikalnych prototypów w narzędziu typu Figma. • Testowanie podstawowych interakcji i przepływów. 6. Podstawy projektowania wizualnego interfejsu: • Ćwiczenia z doboru palety kolorów i typografii. • Projektowanie prostych ikon i elementów UI. • Tworzenie moodboardu i podstawowego UI kitu dla projektu. 7. Projektowanie interfejsu dla konkretnego elementu gry: • Zaprojektowanie (wireframe, prototyp, podstawowy design wizualny) np. menu głównego, HUD, ekranu ekwipunku lub systemu dialogowego. 8. Wprowadzenie do projektowania UI/UX dla VR/AR: • Analiza studiów przypadku interfejsów VR/AR. • Szkicowanie koncepcji interfejsu dla prostej aplikacji VR/AR. 9. Przygotowanie i przeprowadzenie uproszczonych testów użyteczności: • Opracowanie scenariusza testowego dla stworzonego prototypu. • Przeprowadzenie testów (np. z udziałem innych studentów), zbieranie feedbacku. 10. Projekt zespołowy/indywidualny: • Kompleksowe zaprojektowanie (od analizy po interaktywny prototyp i dokumentację) interfejsu użytkownika dla małej gry lub aplikacji interaktywnej.		
17.	Fizyka gier	K_W08, K_W14		K_K01	Wykłady: 1. Wprowadzenie do fizyki w grach komputerowych: • Rola i znaczenie symulacji fizycznych w grach. • Realizm vs. grywalność – kompromisy w fizyce gier. • Przegląd typowych zjawisk fizycznych modelowanych w grach. • Historia i ewolucja fizyki w grach.	EGZ	Egzamin pisemny

					2. Podstawy kinematyki i dynamiki: • Wektory, układy współrzędnych, transformacje. • Ruch prostoliniowy i krzywoliniowy: położenie, prędkość, przyspieszenie. • Prawa Newtona: siła, masa, pęd, moment pędu. • Praca, moc, energia (kinetyczna, potencjalna). Zasada zachowania energii. 3. Detekcja kolizji: • Podstawowe kształty kolizyjne (Bounding Volumes): AABB, OBB, sfery, kapsuły. • Algorytmy detekcji kolizji dla prostych kształtów. • Metody podziału przestrzeni (np. Quadtree/Octree) do optymalizacji detekcji (Broad Phase). • Algorytmy separującej osi (SAT) dla detekcji kolizji wielokątów (Narrow Phase). • Kolizje z terenem i siatkami (meshes). 4. Reakcja na kolizje: • Pojęcie impulsu i jego zastosowanie w reakcji na kolizje. • Współczynnik restytucji (sprężystość zderzeń). • Tarcie (statyczne i kinetyczne). • Rozwiązywanie wielu kontaktów i stabilność symulacji. 5. Dynamika ciał sztywnych (Rigid Body Dynamics): • Ruch obrotowy: prędkość kątowa, przyspieszenie kątowe. • Moment siły (torque), moment bezwładności, tensor bezwładności. • Równania ruchu dla ciał sztywnych. • Modelowanie połączeń (constraints/joints): zawiasy, połączenia kuliste, suwaki. 6. Systemy cząsteczkowe (Particle Systems): • Zasady działania systemów cząsteczkowych. • Modelowanie efektów takich jak ogień, dym, eksplozje, ciecze (podstawowe). • Siły działające na cząsteczki (grawitacja, wiatr, opór). • Cykl życia cząsteczek. 7. Fizyka postaci (Character Physics): • Systemy Ragdoll – symulacja bezwładnego ciała postaci. • Podstawy kinematyki odwrotnej (Inverse Kinematics - IK) dla naturalnego ruchu. • Połączenie animacji z fizyką. 8. Fizyka pojazdów:		

					• Modelowanie uproszczonej fizyki samochodu: siły działające na koła, model opony (np. Pacejka – wprowadzenie). • Zawieszenie, sterowanie, hamowanie. • Podstawy fizyki lotu dla prostych modeli samolotów/dronów. 9. Ciała miękkie i deformowalne (Soft Body Dynamics): • Wprowadzenie do modelowania ciał deformowalnych: systemy masowo-sprężynowe. • Symulacja tkanin (cloth simulation). • Podstawy symulacji płynów (np. SPH – Smoothed Particle Hydrodynamics - wprowadzenie). 10. Metody numeryczne w symulacjach fizycznych: • Całkowanie numeryczne równań ruchu: metoda Eulera, Eulera-Cromera, Verlet, Runge-Kutta (RK4). • Stabilność i dokładność metod numerycznych. • Problem "przenikania" (tunneling) i jego rozwiązywanie. 11. Optymalizacja symulacji fizycznych: • Techniki optymalizacyjne dla detekcji kolizji i symulacji dynamiki. • Wykorzystanie "sleep states" dla obiektów statycznych. • Upraszczanie modeli fizycznych (LOD dla fizyki). 12. Silniki fizyczne: • Przegląd popularnych silników fizycznych (np. PhysX, Bullet Physics, Box2D). • Architektura i główne komponenty silników fizycznych. • Integracja silników fizycznych z silnikami gier (np. Unity, Unreal Engine).		
			K_U01, K_U07, K_U11, K_U12	K_K01	Laboratoria 1. Konfiguracja środowiska i podstawy ruchu: • Implementacja prostego wektora 2D/3D i operacji na wektorach. • Symulacja ruchu obiektu pod wpływem stałej prędkości i przyspieszenia. 2. Implementacja sił i praw Newtona: • Dodawanie sił (np. grawitacja, siła użytkownika) do obiektu. • Symulacja ruchu pod wpływem wypadkowej sił. 3. Prosta detekcja kolizji: • Implementacja detekcji kolizji dla par obiektów: punkt-prostokąt, koło-koło, prostokąt-prostokąt (AABB).		Sprawozdanie z laboratorium, dokumentacja projektowa

					4. Prosta reakcja na kolizje: • Implementacja podstawowej reakcji na kolizję (np. zatrzymanie, odbicie) dla prostych kształtów. 5. Systemy cząsteczkowe: • Tworzenie prostego systemu cząsteczkowego (np. fontanna, deszcz). • Implementacja emiterów, sił i cyklu życia cząsteczek. 6. Dynamika ciał sztywnych – wprowadzenie: • Implementacja ruchu obrotowego prostego obiektu pod wpływem momentu siły (2D). • Eksperymentowanie z momentem bezwładności. 7. Praca z wbudowanym silnikiem fizycznym (np. w Unity/Unreal Engine): • Konfiguracja komponentów Rigidbody i Collider. • Aplikowanie sił i momentów sił za pomocą API silnika. • Tworzenie scen z interakcjami fizycznymi (np. przewracające się klocki, staczająca się kula). 8. Implementacja połączeń (Joints/Constraints): • Wykorzystanie wbudowanych w silnik połączeń (np. Hinge Joint, Spring Joint) do tworzenia złożonych obiektów fizycznych (np. prosty łańcuch, wahadło). 9. Podstawy fizyki pojazdu lub postaci: • Implementacja uproszczonego kontrolera dla pojazdu (ruch, skręcanie z wykorzystaniem fizyki kół). • Konfiguracja podstawowego ragdolla dla postaci. 10. Projekt praktyczny: • Zrealizowanie małego projektu demonstrującego wybrane aspekty fizyki gier (np. prosta gra 2D z fizyką, symulacja zjawiska fizycznego, implementacja specyficznej mechaniki opartej na fizyce).		
18.	Systemy baz danych	K_W09			Wykłady: 1. Pojęcie systemu zarządzania bazą danych (SZDB) i bazy danych, funkcje SZDB, podstawowe metody organizacji danych (model hierarchiczny, sieciowy, relacyjny i obiektowy). 2. Relacyjny model danych: relacje, klucze główne i klucze obce, dziedziny, wartości „null”, elementy algebry relacyjnej. 3. Integralność danych w modelu relacyjnym: integralność encji, referencyjna i dziedziny, metody utrzymywania integralności referencyjnej. 4. Modelowanie bazy danych: normalizacja, postacie normalne, diagramy związków encji.	ZAO	Test pisemny

					5. Wprowadzenie do języka SQL. Podstawowe instrukcje SQL-a. 6. Podzapytania strukturalne i skorelowane		
			K_U24	K_K03	Laboratoria: 1. Polecenia z grupy DDL: tworzenie, modyfikacja i usuwanie tabel, indeksów oraz perspektyw. 2. Polecenia INSERT, UPDATE, DELETE, TRUNCATE. 3. Zapytania proste z użyciem predykatów i sortowania. 4. Złączenia wewnętrzne, zewnętrzne i obustronne. 5. Zapytania z użyciem funkcji agregujących. 6. Podzapytania strukturalne i skorelowane.		Zadania laboratoryjne
19.	Narzędzia AI w projektowaniu gier	K_W10, K_W14		K_K01	Wykłady: 1. Podstawowe koncepcje sztucznej inteligencji i ich znaczenie w projektowaniu gier komputerowych 2. Przegląd narzędzi generujących zasoby graficzne (tekstury, sprite'y) z wykorzystaniem modeli generatywnych 3. Zastosowanie modeli językowych w tworzeniu dialogów i narracji w grach interaktywnych 4. Algorytmy generowania proceduralnego poziomów wspierane przez techniki uczenia maszynowego 5. Metody adaptacyjnego dostosowywania poziomu trudności (Dynamic Difficulty Adjustment) przy użyciu analizy zachowań gracza 6. Wprowadzenie do Unity ML-Agents: architektura, możliwości i przykłady zastosowań w prototypach gier 7. Generatywne sieci neuronowe (GAN, VAE) w kontekście kreowania treści 2D/3D dla gier komputerowych 8. Optymalizacja modeli sztucznej inteligencji w czasie rzeczywistym (quantization, pruning, inference acceleration) 9. Wykorzystanie narzędzi AI do automatycznego testowania i analizy jakości rozgrywki 10. Tworzenie i integracja inteligentnych agentów NPC sterowanych za pomocą technik reinforcement learning 11. Procedury generowania i optymalizacji ścieżek dźwiękowych oraz efektów audio przy użyciu modeli AI 12. Podstawy analizy telemetrii w grach – gromadzenie, przetwarzanie i wykorzystanie danych graczy z użyciem narzędzi AI 13. Aspekty etyczne i prawne związane z korzystaniem z wygenerowanych przez AI treści w produkcji gier	ZAO	Test pisemny

					14. Perspektywy rozwoju sztucznej inteligencji w grach wideo – VR, AR i adaptacyjne środowiska interaktywne		
			K_U10 K_U13	K_K01 K_K04	Laboratoria: 1. Konfiguracja i instalacja narzędzi AI wykorzystywanych w branży gier (Unity ML-Agents, TensorFlow, Stable Diffusion) 2. Generowanie prostych zasobów graficznych 2D przy użyciu modelu Stable Diffusion i import do Unity 3. Tworzenie dialogów NPC z wykorzystaniem modelu językowego (ChatGPT API) w Unity 4. Zbieranie i analiza telemetrii gracza do adaptacji poziomu trudności (eksport CSV + obróbka w Pythonie) 5. Integracja modelu TensorFlow Lite w projekcie Unity (konwersja, import, inferencja) 6. Wprowadzenie do Unity ML-Agents – wdrożenie i uruchomienie przykładowego agenta „3D Ball" 7. Eksperyment z generatywną siecią neuronową (GAN) do kreowania prostych grafik i użycie ich jako assetów w Unity 8. Automatyczne testowanie scenariusza rozgrywki w Unity za pomocą prostych botów symulujących gracza 9. Trenowanie i integracja agenta NPC sterowanego algorytmem DQN (Deep Q-Network) w Unity 10. Generowanie ścieżek dźwiękowych w Google Magenta i synchronizacja ich z silnikiem gry 11. Integracja narzędzia Lobe.ai do klasyfikacji obrazów w czasie rzeczywistym w Unity		3 zadania laboratoryjne
20	Wprowadzenie do silników gier	K_W02, K_W14 K_U01,		K_K01	Wykłady: 1. Co to jest silnik gry? Definicja i cel: • Rola silnika w procesie tworzenia gier. • Zalety korzystania z gotowych silników. • Kiedy warto pisać własny silnik? (krótkie omówienie). • Ewolucja silników gier. 2. Główne komponenty silnika gry: • Ogólna architektura współczesnych silników. • Silnik renderujący (grafika 2D/3D, potok renderowania, oświetlenie, materiały, shadery – wprowadzenie). • Silnik fizyczny (detekcja kolizji, dynamika ciał sztywnych). • System obsługi wejścia (Input). • System audio. • System skryptowy i API. • Zarządzanie zasobami (Assets).	EGZ	Egzamin pisemny

					• Edytor scen/poziomów. • System animacji. • System UI/GUI. • Podstawy networkingu w silnikach. 3. Przegląd popularnych silników gier: • Unity Engine: charakterystyka, mocne strony, typowe zastosowania, model licencjonowania. • Unreal Engine: charakterystyka, mocne strony, typowe zastosowania, model licencjonowania. • Godot Engine: charakterystyka, filozofia open-source, zastosowania. • Inne silniki (np. CryEngine, GameMaker Studio) – krótkie omówienie. 4. Pętla gry (Game Loop): • Koncepcja pętli gry i jej implementacja w silnikach. • Aktualizacja stanu gry, renderowanie, obsługa wejścia. • Stały vs. zmienny krok czasowy (FixedUpdate vs. Update). 5. Paradygmaty projektowe w silnikach: • Model obiektowy i komponentowy (np. Entity-Component-System - ECS). • Scenograf (Scene Graph) i jego rola. 6. Praca z edytorem silnika gry (na przykładzie Unity/Unreal): • Interfejs użytkownika edytora: okna, narzędzia, nawigacja. • Zarządzanie projektami i scenami. • Hierarchia obiektów. • Inspektor właściwości. 7. Podstawy tworzenia scen i poziomów: • Obiekty gry (Game Objects / Actors). • Komponenty i ich rola. • Transformacje: pozycja, rotacja, skala. • Systemy współrzędnych (globalne, lokalne). • Praca z kamerą. 8. Zarządzanie zasobami (Assets): • Importowanie modeli 3D, tekstur, dźwięków, animacji. • Organizacja zasobów w projekcie. • Prefaby (Prefabs / Blueprints) jako szablony obiektów. 9. Wprowadzenie do skryptowania w silnikach (np. C# w Unity, Blueprints/C++ w Unreal): • Podstawy logiki gry. • Obsługa zdarzeń (np. kolizje, kliknięcia myszy). • Komunikacja między obiektami/skryptami.		

					• Debugowanie skryptów. 10. Grafika i materiały w silnikach: • Podstawy pracy z materiałami i shaderami (wprowadzenie). • Teksturowanie obiektów. • Oświetlenie sceny (światła kierunkowe, punktowe, ambientowe). • Efekty post-processingu (wprowadzenie). 11. Fizyka w silnikach: • Konfiguracja ciał sztywnych (Rigidbodies). • Kolizje i wyzwalacze (Colliders, Triggers). • Stosowanie sił i momentów sił. • Połączenia (Joints). 12. Budowanie i wdrażanie gry: • Proces kompilacji projektu dla różnych platform (PC, mobile – wprowadzenie). • Ustawienia projektu i optymalizacja (wprowadzenie). • Testowanie gry.		
			K_U01, K_U10, K_U11, K_U13, K_U22	K_K01, K_K05	Laboratoria: 1. Instalacja i konfiguracja wybranego silnika gry (np. Unity lub Unreal Engine): • Pobieranie instalatora, proces instalacji. • Tworzenie nowego projektu. 2. Zapoznanie z interfejsem edytora silnika: • Nawigacja po głównych oknach (Scene, Hierarchy, Project, Inspector). • Podstawowe narzędzia manipulacji obiektami. 3. Tworzenie prostej sceny 2D/3D: • Dodawanie podstawowych obiektów (sześciany, kule, płaszczyzny / sprite'y). • Ustawianie pozycji, rotacji i skali obiektów. • Konfiguracja kamery. 4. Importowanie i zarządzanie zasobami: • Importowanie prostych modeli 3D, tekstur, plików audio. • Tworzenie folderów i organizacja zasobów w panelu projektu. 5. Praca z materiałami i teksturami: • Tworzenie i przypisywanie prostych materiałów do obiektów. • Aplikowanie tekstur. 6. Podstawy skryptowania – ruch obiektu:		Zadana laboratoryjne, dokumentacja projektowa

					• Tworzenie pierwszego skryptu (np. C# w Unity, prosty Blueprint w Unreal). • Implementacja ruchu obiektu sterowanego przez gracza (np. klawiatura) lub automatycznego. 7. Podstawy fizyki – kolizje: • Dodawanie komponentów Collider i Rigidbody do obiektów. • Obserwacja interakcji fizycznych (np. spadanie, odbijanie się). • Implementacja prostej reakcji na kolizję (np. zniszczenie obiektu). 8. Tworzenie prefabów/blueprintów: • Tworzenie szablonu obiektu (np. przeciwnik, moneta) do wielokrotnego użytku. • Instancjonowanie prefabów w scenie. 9. Podstawy UI: • Tworzenie prostego interfejsu użytkownika (np. licznik punktów, przycisk). 10. Budowanie i testowanie prostego projektu: • Proces budowania projektu do postaci wykonywalnej. • Testowanie podstawowej funkcjonalności. 11. Mini-projekt: • Stworzenie bardzo prostej gry lub interaktywnej sceny wykorzystującej poznane elementy silnika (np. "zbieracz monet", prosta platformówka 2D, scena z interaktywnymi obiektami fizycznymi).		
21.	Projekt indywidualny – gra 2d	K_W02, K_W06, K_W10, K_W11, K_W12, K_W13, K_W14, K_W16		K_K01	Wykłady: 1. Wybór i zdefiniowanie koncepcji gry 2D: • Burza mózgów, generowanie pomysłów. • Określenie gatunku gry (np. platformówka, zręcznościowa, logiczna, przygodowa typu point-and-click, shoot'em up, itp.). • Definicja głównej mechaniki (core gameplay loop). • Określenie grupy docelowej. • Analiza wykonalności pomysłu w ramach dostępnego czasu i umiejętności. • Produkt: Krótki dokument koncepcyjny (1-2 strony) zatwierdzany przez prowadzącego. • (Sposób weryfikacji etapu): Ocena dokumentu koncepcyjnego, konsultacje. 2. Projektowanie gry (Preprodukcja):	ZAO	Test pisemny

					• Stworzenie uproszczonego Game Design Document (GDD) dla projektu, zawierającego: ▪ Opis fabuły (jeśli dotyczy) i świata gry. ▪ Szczegółowy opis mechanik gry. ▪ Projekt poziomów (szkice, koncepcje). ▪ Projekt postaci i przeciwników (wygląd, zachowania). ▪ Projekt interfejsu użytkownika (UI/UX). ▪ Wymagania dotyczące grafiki i dźwięku. • Wybór technologii/narzędzi do tworzenia gry 2D: ▪ oparte na bibliotekach niskopoziomowych (np. C++ z SDL/SFML/Allegro) lub C# .NET z odpowiednimi bibliotekami graficznymi. ▪ oparte na frameworkach/bibliotekach JavaScript (np. Phaser.js, PixiJS). ▪ oparte na silnikach gier z wsparciem dla 2D (np. Unity 2D, Godot Engine, GameMaker Studio – z uwzględnieniem, że jest to projekt wprowadzający do samodzielnego tworzenia, więc wybór silnika powinien być świadomy i uzasadniony przez studenta, potencjalnie po wcześniejszym doświadczeniu z bibliotekami). ▪ Wybór powinien być uzgodniony z prowadzącym i dostosowany do umiejętności oraz celów projektowych studenta. • Planowanie pracy, stworzenie harmonogramu realizacji projektu (kamienie milowe). • Produkt: Uproszczony GDD, harmonogram. • (Sposób weryfikacji etapu): Ocena GDD, konsultacje, przegląd harmonogramu		
			K_U03, K_U04, K_U06, K_U10, K_U12, K_U13, K_U15, K_U16, K_U19, K_U20, K_U22, K_U25, K_U26	K_K01, K_K04, K_K05	Laboratorium: 1. Implementacja i tworzenie zasobów (Produkcja): • Programowanie (zależnie od wybranej technologii): ▪ Implementacja głównej logiki gry i mechanik (np. pętla gry, zarządzanie stanem, obsługa wejścia, renderowanie grafiki 2D, obsługa dźwięku). ▪ Tworzenie systemu sterowania postacią/obiektami. ▪ Implementacja AI przeciwników (jeśli dotyczy). ▪ System kolizji (prosta detekcja i reakcja, np. AABB) i podstawowej fizyki (jeśli wymagany). ▪ Implementacja interfejsu użytkownika.		Projekt gry

					▪ Zarządzanie stanem gry (np. zapis/odczyt – opcjonalnie, w zależności od zakresu). • Grafika: ▪ Tworzenie lub pozyskiwanie (z poszanowaniem licencji) grafiki 2D (postacie, tła, elementy interfejsu, efekty). ▪ Animacja postaci i obiektów (sprite sheets, animacja klatkowa/szkieletowa 2D). • Dźwięk: ▪ Tworzenie lub pozyskiwanie (z poszanowaniem licencji) efektów dźwiękowych i muzyki. ▪ Implementacja dźwięku w grze. • Projektowanie poziomów: ▪ Budowa co najmniej jednego grywalnego poziomu (lub kilku mniejszych, w zależności od koncepcji). • Regularne konsultacje z prowadzącym, prezentacja postępów. • Iteracyjny rozwój, testowanie i poprawianie błędów na bieżąco. • Produkt: Działający prototyp gry, kolejne iteracje gry, zasoby graficzne i dźwiękowe. • (Sposób weryfikacji etapu): Regularne przeglądy kodu i grywalnych wersji, ocena postępów względem harmonogramu, konsultacje. 2. Testowanie i dopracowywanie: • Systematyczne testowanie gry pod kątem błędów (bugów) i problemów z grywalnością. • Zbieranie feedbacku (np. od kolegów, prowadzącego). • Balansowanie rozgrywki. • Optymalizacja gry (jeśli konieczne). • Dopracowanie grafiki, dźwięku, interfejsu. • Produkt: Wersja beta gry. • (Sposób weryfikacji etapu): Ocena grywalności, raporty z testów, konsultacje. 3. Finalizacja i prezentacja projektu: • Przygotowanie finalnej, grywalnej wersji gry. • Stworzenie krótkiej dokumentacji technicznej i projektowej (podsumowującej GDD, wybór technologii i kluczowe aspekty implementacyjne, proces tworzenia). • Przygotowanie prezentacji projektu.		

					• Publiczna (w ramach grupy/kierunku) prezentacja gry, omówienie zastosowanych rozwiązań, napotkanych problemów i wniosków. • Produkt: Finalna wersja gry 2D, dokumentacja, prezentacja. • (Sposób weryfikacji etapu / Zaliczenie przedmiotu): Ocena finalnego produktu (grywalność, kompletność, zgodność z założeniami, jakość techniczną i artystyczną w ramach indywidualnych możliwości oraz wybranej technologii), ocena dokumentacji, ocena prezentacji projektu.		
22.	Technologie immersyjne	K_W02, K_W10, K_W14, K_W16		K_K01	Wykłady: 1. Wprowadzenie do technologii immersyjnych: • Definicje Wirtualnej Rzeczywistości (VR), Rozszerzonej Rzeczywistości (AR) i Mieszanej Rzeczywistości (MR/XR). Kluczowe różnice i podobieństwa. • Historia rozwoju technologii VR/AR. • Pojęcie immersji i obecności (presence). • Spektrum wirtualności (Virtuality Continuum) Milgrama. 2. Sprzęt i platformy VR: • Rodzaje gogli VR: PC VR (np. Valve Index, HTC Vive, Oculus Rift S), Standalone VR (np. Meta Quest, Pico Neo), Mobile VR (np. Google Cardboard – historycznie). • Technologie wyświetlania (OLED, LCD), rozdzielczość, pole widzenia (FOV), częstotliwość odświeżania. • Systemy śledzenia (tracking): inside-out, outside-in, 3DoF vs 6DoF. • Kontrolery VR i inne metody interakcji (śledzenie rąk, haptyka). • Platformy deweloperskie (np. SteamVR, Oculus SDK, OpenXR). 3. Sprzęt i platformy AR: • Rodzaje urządzeń AR: smartfony i tablety (ARKit, ARCore), okulary AR (np. Microsoft HoloLens, Magic Leap, Nreal). • Technologie wyświetlania w AR (np. waveguide, birdbath optics). • Metody śledzenia w AR: marker-based, markerless (SLAM - Simultaneous Localization and Mapping), GPS-based. • Interakcja w AR: dotyk, gesty, głos. • Platformy deweloperskie (np. ARKit, ARCore, Vuforia, MRTK - Mixed Reality Toolkit). 4. Projektowanie interakcji w VR (VR Interaction Design): • Specyfika projektowania dla VR – odejście od tradycyjnych paradygmatów 2D.	EGZ	Egzamin pisemny

					• Naturalne interakcje: chwytanie, rzucanie, manipulowanie obiektami. • Metody lokomocji w VR (teleportacja, smooth locomotion, arm-swinger) i ich wpływ na komfort. • Projektowanie interfejsów użytkownika (UI) w przestrzeni 3D. • Ergonomia i komfort użytkownika – zapobieganie chorobie symulatorowej (cybersickness). 5. Projektowanie interakcji w AR (AR Interaction Design): • Kontekstualność informacji i interakcji w AR. • Nakładanie treści cyfrowych na świat rzeczywisty. • Rozpoznawanie obiektów i otoczenia. • Interfejsy użytkownika dostosowane do AR (np. umieszczanie UI na powierzchniach rzeczywistych). • Wyzwania związane z okluzją (occlusion). 6. Tworzenie treści dla VR/AR: • Modelowanie 3D i optymalizacja dla VR/AR (polygony, tekstury, draw calls). • Tworzenie materiałów i oświetlenia zoptymalizowanego pod kątem wydajności. • Dźwięk przestrzenny (Spatial Audio) i jego rola w budowaniu immersji. • Wykorzystanie fotogrametrii i skanowania 3D do tworzenia zasobów. • Narzędzia i silniki do tworzenia aplikacji VR/AR (np. Unity, Unreal Engine z odpowiednimi SDK). 7. Zastosowania technologii VR: • Gry i rozrywka. • Szkolenia i symulacje (np. medyczne, przemysłowe, wojskowe). • Edukacja i wirtualne wycieczki. • Architektura i wizualizacje projektów. • Terapia i rehabilitacja. • Wirtualne spotkania i współpraca (Social VR). 8. Zastosowania technologii AR: • Gry mobilne (np. Pokémon GO). • Marketing i reklama (interaktywne produkty, wirtualne przymierzalnie). • Przemysł (wsparcie produkcji, serwis, instrukcje). • Edukacja i muzea (interaktywne eksponaty). • Nawigacja i informacja w przestrzeni.		

					• Medycyna (wsparcie operacji, wizualizacja danych). 9. Wyzwania i ograniczenia technologii immersyjnych: • Koszt sprzętu i dostępność. • Problemy z komfortem użytkowania (choroba symulatorowa, zmęczenie oczu). • Ograniczenia techniczne (FOV, rozdzielczość, moc obliczeniowa). • Kwestie bezpieczeństwa i prywatności. • Brak standardów i fragmentacja rynku. 10. Etyczne i społeczne aspekty VR/AR: • Wpływ na percepcję rzeczywistości. • Uzależnienia i izolacja społeczna. • Prywatność danych w świecie rzeczywistym i wirtualnym. • Potencjał manipulacji i dezinformacji. 11. Przyszłość technologii immersyjnych: • Trendy rozwojowe (miniaturyzacja, poprawa jakości obrazu, bardziej naturalne interakcje). • Integracja z AI i IoT. • Rozwój Metaverse. • Nowe obszary zastosowań. 12. Praktyczne aspekty tworzenia aplikacji VR/AR w silnikach gier: • Konfiguracja projektu w Unity/Unreal Engine dla VR/AR. • Praca z SDK (Oculus, SteamVR, ARKit, ARCore, MRTK). • Implementacja podstawowych mechanik interakcji. • Testowanie i debugowanie na urządzeniach docelowych.		
			K_U01, K_U13, K_U14, K_U18, K_U22	K_K01, K_K02, K_K05	Laboratoria: 1. Konfiguracja środowiska deweloperskiego dla VR/AR: • Instalacja silnika gry (Unity/Unreal Engine) i odpowiednich SDK. • Konfiguracja projektu pod kątem wybranej platformy VR lub AR. • Podłączenie i konfiguracja sprzętu (gogle VR, smartfon z AR). 2. Tworzenie prostej sceny VR: • Ustawienie kamery VR, kontrolerów. • Implementacja podstawowej lokomocji (np. teleportacja). • Interakcja z prostymi obiektami w scenie (np. podnoszenie, rzucanie). 3. Tworzenie prostej aplikacji AR: • Detekcja płaszczyzn i umieszczanie obiektów 3D w świecie rzeczywistym (markerless AR).		Zadania laboratoryjne, dokumentacja projektowa

					• Prosta interakcja z obiektami AR (np. skalowanie, obracanie dotykiem). • (Opcjonalnie) Wykorzystanie markerów do wyświetlania treści AR. 4. Projektowanie i implementacja UI dla VR: • Tworzenie interfejsu użytkownika widocznego w przestrzeni 3D (np. menu, wskaźniki). • Interakcja z UI za pomocą kontrolerów VR (np. raycasting). 5. Projektowanie i implementacja UI dla AR: • Umieszczanie elementów UI w kontekście sceny AR. • Dostosowanie interfejsu do interakcji dotykowych lub gestów. 6. Optymalizacja zasobów 3D dla VR/AR: • Analiza modeli pod kątem liczby polygonów i tekstur. • Techniki optymalizacji (np. LOD, texture baking – wprowadzenie). 7. Implementacja dźwięku przestrzennego w VR: • Dodawanie źródeł dźwięku 3D do sceny. • Testowanie efektu immersji dźwiękowej. 8. Praca z systemami śledzenia rąk (jeśli dostępne): • Implementacja prostych interakcji opartych na gestach dłoni. 9. Wykorzystanie Mixed Reality Toolkit (MRTK) lub podobnych narzędzi: • Tworzenie aplikacji z wykorzystaniem gotowych komponentów i narzędzi MRTK. 10. Testowanie i debugowanie aplikacji VR/AR na urządzeniach: • Techniki profilowania wydajności. • Identyfikacja i rozwiązywanie problemów specyficznych dla VR/AR (np. judder, problemy ze śledzeniem). 11. Mini-projekt VR lub AR: • Stworzenie małej, interaktywnej aplikacji VR lub AR demonstrującej poznane techniki (np. wirtualna galeria, prosta gra AR, interaktywna wizualizacja produktu). Prezentacja projektu.		
23a	Przedsiębiorczość	K_W19			Wykłady: 1. Przemysł 1.0-4.0, rozumienia pojęcie przedsiębiorczość oraz rodzaje przedsiębiorczości i ich charakterystyka 2. I forma przedsiębiorczości – przedsiębiorczość w ujęciu funkcjonalnym, znaczenie sektora przedsiębiorstw MŚP w gospodarce Polskiej i światowej. Wpływ i znaczenie innowacyjności oraz Badań i Rozwoju na przedsiębiorczość,	ZAO	Opracowanie przez studentów projektu w postaci prezentacji multimedialnej z

					wybrane programy wsparcia w różnych obszarach aktywności firm z sektora MŚP 3. II forma przedsiębiorczości – przedsiębiorca jako jednostka „uprawiająca" przedsiębiorczość, cechy i składowe profilu przedsiębiorcy 4. III forma przedsiębiorczości – przedsiębiorczość jako podejście do zarządzania, wybrane aspekty ekonomiczne funkcjonowania przedsiębiorstwa z sektora MŚP, identyfikacja i ocena szans, zagrożeń i ryzyka. Bariery w tworzeniu i funkcjonowaniu firm z sektora MŚP, innowacyjność podstawą działań przedsiębiorczych, źródła wsparcia przedsiębiorców w ich działaniu		wybranego zagadnienia z listy, dotyczącego szeroko rozumianego pojęcia przedsiębiorczości
			K_U02, K_U20	K_K02	Ćwiczenia: 1. Podstawy prawne i zasady podejmowania i prowadzenia działalności gospodarczej w Polsce. Zwięzłe przedstawienie regulacji prawnych normujących podejmowanie i prowadzenie działalności gospodarczej w oparciu o przepisy ustawy z dn. 6 marca 2018 r. Prawo przedsiębiorców. 2. Formy organizacyjno-prawne działalności gospodarczej: Prezentacja form organizacyjno- prawnych działalności gospodarczej oraz podstawowych warunków koniecznych do podjęcia działalności, 3. Procedura rejestracji działalności gospodarczej przez osoby fizyczne – Centralna Ewidencja i Informacja o Działalności Gospodarczej a. Zasady działania CEiDG b. Procedura rejestracji w formie tradycyjnej i w trybie elektronicznym c. Rejestracja spółki w Krajowym Rejestrze Sadowym - d. Zasady działania KRS i postępowanie rejestrowe e. Formularze rejestrowe - wypełnianie wybranych formularzy – ćwiczenia 4. Publicznoprawne obowiązki przedsiębiorcy względem organów administracji publicznej –Formy opodatkowania.		Realizacja zadań dotyczących założenia i funkcjonowania firmy
			K_U22	K_K02	Laboratoria: Branżowe Symulacje Biznesowe – Planowanie Biznesu. Zadaniem studenta jest założenie jednoosobowej działalności gospodarczej do wyboru w różnych branżach i planowanie rozwoju firmy. Gra ocenia uczestnika na każdym etapie gry pod kątem podjętych decyzji, sprawdza ich poprawność, spójność, logiczność oraz zgodność z prawem		Uczestnictwo i ukończenie gry przy wymogu uzyskania minimum 50% punktów w każdej rundzie gry symulacyjnej.

23b	Zarządzanie firmą				Wykłady: 1. Proces powstawania przedsiębiorstw 2. Formy prawne przedsiębiorstw 3. Poziomy zarządzania przedsiębiorstwem 4. Zasoby i systemy zarządzania 5. Specyfika przedsiębiorstw usługowych i produkcyjnych 6. Przedsiębiorstwa procesowe i projektowe 7. Źródła finansowania 8. Biznes plan	ZAO	Test pisemny
		K_W19	K_U02, K_U20	K_K02	Ćwiczenia: 1. Orientacja marketingowa firmy i jej wyznaczniki 2. System informacji marketingowej 3. Segmentacja rynków i wybór rynku docelowego 4. Strategia produktu: 5. Strategia cenowa 6. Strategia promocji 7. Strategia dystrybucji 8. Odmiany współczesnego marketingu • Marketing stałych relacji • Marketing internetowy • Marketing partyzancki 9. Procesy logistyczne w przedsiębiorstwie • Procesy logistyczne zaopatrzenia i dystrybucji • Wewnętrzne procesy logistyczne		Poprawność udzielonych odpowiedzi w trakcie dyskusji
			K_U22	K_K02	Laboratoria: 1. Wprowadzenie do koncepcji gry 2. Zapoznanie się z dokumentacją gry 3. Tworzenie wirtualnej firmy 4. Wstępne rozgrywki 5. Rozgrywki w zespołach 4-osobowych 6. Samoocena rozgrywki		Wyniki rozgrywki
24.	Programowanie w silnikach gier	K_W02, K_W14		K_K01	Wykłady: 1. Zaawansowane wprowadzenie do architektury silników Unity i Unreal Engine: • Porównanie filozofii i architektury obu silników. • Główne systemy i moduły (renderowanie, fizyka, audio, UI, animacja). • Pętla gry i zarządzanie cyklem życia obiektów/aktorów. • Model obiektowy vs. komponentowy (Entity-Component-System w Unity, Actor-Component w Unreal). 2. Programowanie w C# dla Unity:	EGZ	Egzamin pisemny

					• Przypomnienie i rozszerzenie kluczowych koncepcji C# istotnych w Unity (klasy, obiekty, dziedziczenie, interfejsy, zdarzenia, delegaty). • Struktura skryptów MonoBehaviour: cykl życia (Awake, Start, Update, FixedUpdate, LateUpdate, OnDestroy), Coroutines. • Komunikacja między skryptami: GetComponent, SendMessage, zdarzenia, referencje publiczne. • Zarządzanie sceną i obiektami z poziomu kodu. • Praca z prefabami i ich instancjonowanie. • Serializacja danych i atrybuty (SerializeField, HideInInspector). 3. Programowanie w C++ i Blueprints dla Unreal Engine: • Wprowadzenie do C++ w kontekście Unreal Engine: składnia, zarządzanie pamięcią (UObject, Garbage Collection), makra (UPROPERTY, UFUNCTION, UCLASS). • Struktura klas Unreal Engine: Actor, Pawn, Character, Controller, GameMode, GameState, PlayerState. • Komponenty (Actor Components, Scene Components) i ich tworzenie oraz wykorzystanie. • Komunikacja między obiektami: referencje, delegaty, interfejsy, Event Dispatchers. • Blueprints: tworzenie logiki wizualnej, integracja z C++, typy Blueprintów (Actor, Function Library, etc.). • Zalety i wady programowania w C++ vs. Blueprints, kiedy używać którego podejścia. 4. Zarządzanie wejściem (Input): • Obsługa klawiatury, myszy, gamepada. • Systemy mapowania wejścia (Input Manager w Unity, Enhanced Input w Unreal). 5. Fizyka i detekcja kolizji: • Praca z komponentami Rigidbody/PrimitiveComponent i Collider/CollisionComponent. • Rodzaje kolizji (statyczne, dynamiczne, kinematyczne) i warstwy kolizji. • Promienie: Raycasting, Spherecasting, Boxcasting i ich zastosowania (np. interakcja, wykrywanie podłoża). • Praca z połączeniami (Joints/Constraints). 6. Systemy animacji w silnikach: • Unity: Animator Controller, stany animacji, przejścia, parametry, warstwy, maski awatara, root motion, Animation Events, programowalne sterowanie animacjami.		

					• Unreal Engine: Animation Blueprints, State Machines, Blend Spaces. • Root motion - programowalne sterowanie animacjami. • Animation Events – wywoływanie funkcji z animacji. • Kinematyka odwrotna (IK) i jej implementacja. 7. Programowanie interfejsu użytkownika (UI): • Unity: Canvas, praca z komponentami UI (Text, Image, Button), system zdarzeń UI, kotwiczenie i skalowanie, tworzenie dynamicznych interfejsów. • Unreal Engine: UMG (Unreal Motion Graphics), Widget Blueprints, praca z podstawowymi widżetami, wiązanie danych (Property Binding), zdarzenia UI, animacja UI. 8. Zarządzanie danymi i stanem gry: • Zapisywanie i wczytywanie postępów gry (np. PlayerPrefs w Unity, SaveGame object w Unreal). • Serializacja i deserializacja danych (np. JSON, XML, formaty binarne). • Wykorzystanie ScriptableObjects (Unity) / Data Assets (Unreal) do zarządzania konfiguracją i danymi. • Podstawy wzorców projektowych w programowaniu gier (np. Singleton, State, Observer). 9. Struktura projektu i dobre praktyki: • Organizacja zasobów i skryptów w projekcie. • Nazewnictwo zmiennych, funkcji, klas. • Komentowanie kodu. • Podstawy debugowania w edytorze silnika. 10. Tworzenie prostych mechanik gry: • Integracja poznanych systemów do tworzenia kompletnych, choć prostych, mechanik (np. system zdrowia, zbieranie przedmiotów, proste zadania). 11. Budowanie i wdrażanie projektu: • Podstawy procesu budowania gry dla wybranej platformy (np. PC). • Podstawowe ustawienia projektu związane z budowaniem.		
			K_U01, K_U03, K_U11, K_U13, K_U16, K_U22, K_U25	K_K01, K_K05	Laboratoria: 1. Konfiguracja środowiska i pierwszy skrypt: • Tworzenie projektu, zapoznanie się ze strukturą. • Napisanie i przetestowanie pierwszego skryptu sterującego obiektem. 2. Implementacja sterowania postacią gracza: • Ruch postaci (2D/3D) za pomocą klawiatury/gamepada.		Zadanie laboratoryjne., dokumentacja projektowa

					• Implementacja akcji w ruchu (skok / ślizg). 3. Podstawowa interakcja z otoczeniem: • Detekcja kolizji z obiektami. • Interakcja z otoczeniem (np. podnoszenie przedmiotów). 4. Tworzenie dynamicznego interfejsu użytkownika: • Implementacja HUD, menu, ekwipunku. • Wiązanie danych UI ze stanem gry. • Obsługa przycisków. 5. Praca z systemem animacji: • Podłączenie animacji do postaci. • Sterowanie przejściami między animacjami z poziomu kodu w odpowiedzi na akcje gracza. 6. Implementacja prostego systemu fizyki: • Konfiguracja obiektów fizycznych. • Wykorzystanie sił do poruszania obiektami. 7. Zarządzanie stanem gry i zapis/odczyt: • Implementacja systemu zapisywania i wczytywania postępów. 8. Integracja dźwięku: • Odtwarzanie efektów dźwiękowych w odpowiedzi na zdarzenia w grze. 9. Tworzenie przeciwnika AI: • Ruch patrolowy NPC lub podążanie za graczem. • Reakcja NPC na gracza. 10. Projekt (prosta gra lub zaawansowana mechanika na bazie nabytych umiejętności): • Zastosowanie poznanych technik programistycznych do stworzenia działającego prototypu w wybranym silniku. Prezentacja i omówienie kodu.		
25.	Tworzenie gier na urządzenia mobilne	K_W02, K_W13		K_K01	Wykłady: 1. Wprowadzenie do tworzenia gier mobilnych: • Rynek gier mobilnych: trendy, platformy. • Specyfika urządzeń mobilnych: ograniczenia sprzętowe (CPU, GPU, pamięć, bateria), różnorodność ekranów, metody wprowadzania danych (dotyk, akcelerometr, żyroskop). • Podejścia do tworzenia gier mobilnych: natywne (Android SDK), silniki (Unity/Unreal), webowe (HTML5/JavaScript), hybrydowe. 2. Podstawy platformy Android dla deweloperów gier: • Architektura systemu Android.	ZAO	Test.

					• Narzędzia deweloperskie: Android Studio, Android SDK, Emulator (AVD), debugowanie na urządzeniu fizycznym. • Cykl życia aplikacji Android i jego znaczenie dla gier. • Formaty dystrybucyjne: APK, App Bundle. 3. Tworzenie gier mobilnych w silniku Unity (dla Androida): • Konfiguracja projektu Unity pod kątem platformy Android. • Obsługa wejścia mobilnego: dotyk (pojedynczy, wielokrotny), gesty, akcelerometr. • Dostosowywanie interfejsu użytkownika (UI) do różnych rozdzielczości i proporcji ekranu (Canvas Scaler, Anchors). • Optymalizacja wydajności dla urządzeń mobilnych w Unity (podstawy – szczegóły w przedmiocie 31a). • Budowanie i wdrażanie gry na emulatorze i urządzeniu Android. 4. Wprowadzenie do tworzenia gier przeglądarkowych w JavaScript: • Podstawy HTML5 i CSS w kontekście gier. • Kluczowe elementy JavaScript dla tworzenia gier (DOM, zdarzenia, pętle, funkcje, obiekty). • Canvas API do rysowania grafiki 2D. 5. Biblioteka Phaser.js – wprowadzenie i podstawy: • Czym jest Phaser.js i dlaczego warto go używać. • Konfiguracja środowiska deweloperskiego dla Phaser.js. • Struktura projektu w Phaser: sceny (Scenes), ładowanie zasobów (preload), tworzenie obiektów (create), pętla gry (update). • Podstawowe elementy: sprite'y, tekst, grupy. 6. Tworzenie mechanik gry w Phaser.js: • Obsługa wejścia (mysz, klawiatura, dotyk). • Fizyka w Phaser.js (Arcade Physics). • Detekcja kolizji. • Animacje klatkowe. • Dźwięk i muzyka. 7. Gry hybrydowe – koncepcja i narzędzia: • Czym są aplikacje hybrydowe (WebView). • Zalety i wady podejścia hybrydowego. • Przegląd narzędzi do tworzenia aplikacji hybrydowych (np. Apache Cordova, Capacitor – omówienie koncepcji bez głębokiego wchodzenia w każde z nich). • Jak opakować grę Phaser.js w aplikację mobilną (np. za pomocą Cordova CLI).		

					8. Monetyzacja gier mobilnych (wprowadzenie): • Modele monetyzacji: premium, freemium, reklamy (bannery, pełnoekranowe, nagradzane), zakupy w aplikacji (IAP). • Podstawy integracji reklam (np. Google AdMob – koncepcja). 9. Proces publikacji gry w sklepie Google Play: • Założenie konta deweloperskiego Google Play Console. • Przygotowanie materiałów do publikacji: ikona, zrzuty ekranu, opis, polityka prywatności. • Konfiguracja aplikacji w konsoli: nazwa, kategoria, ocena treści. • Przesyłanie i zarządzanie plikami APK/App Bundle. • Testowanie wewnętrzne, alfa, beta. • Wydawanie gry do produkcji. 10. Aspekty projektowania gier mobilnych: • User Experience (UX) na urządzeniach mobilnych. • Projektowanie interfejsu (UI) pod kątem dotyku i małych ekranów. • Sesje gracza w grach mobilnych.		
			K_U01, K_U03, K_U10, K_U11, K_U13, K_U16, K_U25	K_K01, K_K04, K_K05	Laboratoria: 1. Konfiguracja środowiska Android i Unity: • Instalacja Android Studio i SDK. Konfiguracja emulatora AVD. • Podłączenie urządzenia fizycznego do debugowania. • Konfiguracja projektu Unity dla Androida, build i uruchomienie pustej sceny. 2. Podstawowa gra mobilna w Unity: • Implementacja sterowania dotykowego postacią/obiektem. • Stworzenie prostego UI skalującego się na różne ekrany. • Testowanie na emulatorze i urządzeniu. 3. Wprowadzenie do Phaser.js – pierwsza scena: • Konfiguracja lokalnego serwera deweloperskiego. • Stworzenie podstawowej struktury gry w Phaser.js, załadowanie zasobów, wyświetlenie sprite'a. 4. Mechaniki gry w Phaser.js – ruch i interakcja: • Implementacja sterowania postacią (początkowo klawiatura dla ułatwienia, potem dotyk). • Implementacja prostych kolizji z wykorzystaniem Arcade Physics. 5. Mechaniki gry w Phaser.js – UI i logika gry:		Zadanie projektowe

					• Dodanie punktacji, prostego menu, ekranu końca gry. • Implementacja obsługi dotyku dla interfejsu i rozgrywki. 6. Tworzenie responsywnej gry w Phaser.js: • Dostosowanie logiki gry i UI do różnych rozmiarów okna przeglądarki/ekranu. 7. Budowanie gry hybrydowej (Phaser.js + Cordova/Capacitor – uproszczone): • Instalacja narzędzia (np. Cordova). • Utworzenie projektu hybrydowego i zintegrowanie gry Phaser.js. • Skompilowanie i uruchomienie aplikacji na emulatorze/urządzeniu Android. 8. Implementacja zapisu postępu (lokalnie): • Wykorzystanie PlayerPrefs (Unity) lub LocalStorage (JavaScript) do zapisu np. najlepszego wyniku. 9. Przygotowanie gry do publikacji: • Stworzenie ikony aplikacji i zrzutów ekranu. • Wygenerowanie podpisanego pliku APK/App Bundle. 10. Projekt – Stworzenie i "opublikowanie" prostej gry mobilnej: • Student tworzy (indywidualnie lub w małej grupie) prostą grę na platformę Android (w Unity lub Phaser.js jako aplikacja hybrydowa). • Przygotowuje wszystkie niezbędne materiały do symulowanej publikacji w Google Play (opis, grafiki). • Prezentacja gry i procesu "publikacji".		
26.	Aspekty marketingowe branż gier	K_W02, K_W19, K_W20,		K_K01	Wykłady: 1. Wprowadzenie do marketingu gier komputerowych: • Definicja marketingu i jego rola w cyklu życia gry. • Specyfika rynku gier: segmentacja, trendy, platformy. • Marketing mix (4P/7P) w kontekście gier (Produkt, Cena, Dystrybucja, Promocja). • Różnice w marketingu gier indie vs. AAA. 2. Badania rynku i analiza grupy docelowej: • Metody badań rynkowych (ilościowe, jakościowe). • Identyfikacja i charakterystyka grupy docelowej (target audience). • Tworzenie person graczy. • Analiza konkurencji i Unique Selling Proposition (USP) gry. 3. Branding i pozycjonowanie gry: • Budowanie marki gry: nazwa, logo, identyfikacja wizualna. • Komunikacja wartości i pozycjonowanie gry na rynku.	ZAO	Test

					• Storytelling w marketingu gier. 4. Kanały dystrybucji gier: • Platformy cyfrowe (Steam, Epic Games Store, GOG, konsole, sklepy mobilne). • Specyfika marketingu na poszczególnych platformach. • Wydawcy gier – rola i współpraca. Self-publishing. 5. Public Relations (PR) w branży gier: • Rola PR w budowaniu wizerunku i relacji z mediami. • Tworzenie materiałów prasowych (press kit, notatki prasowe). • Współpraca z dziennikarzami, blogerami, influencerami i mediami branżowymi. • Zarządzanie kryzysowe w PR. 6. Influencer Marketing: • Identyfikacja i wybór odpowiednich influencerów (streamerzy, YouTuberzy, recenzenci). • Modele współpracy z influencerami. • Mierzenie efektywności kampanii influencerskich. 7. Community Management i Social Media Marketing: • Budowanie i zarządzanie społecznością graczy (Discord, fora, Reddit). • Wykorzystanie mediów społecznościowych (Facebook, Twitter, Instagram, TikTok, YouTube) w promocji gier. • Tworzenie angażujących treści, prowadzenie kampanii w social media. 8. Content Marketing i materiały promocyjne: • Tworzenie wartościowych treści (devlogi, artykuły, poradniki). • Produkcja materiałów wideo: trailery (teaser, gameplay, launch), gameplaye. • Optymalizacja strony gry w sklepach (store page optimization – Steam, Google Play etc.). 9. Reklama płatna w promocji gier: • Podstawy reklamy cyfrowej (PPC, reklama w social mediach, reklama na platformach gamingowych). • Targetowanie reklam, budżetowanie kampanii. • Narzędzia analityczne do mierzenia efektywności reklam. 10. Marketing przedpremierowy (Pre-launch): • Budowanie świadomości i "hype'u" przed premierą. • Zbieranie wishlist, budowanie listy mailingowej. • Organizowanie testów (alfa, beta) jako element promocji.		

					11. Marketing premierowy (Launch) i popremierowy (Post-launch): • Strategie na dzień premiery. • Utrzymanie zainteresowania grą po premierze: aktualizacje, DLC, eventy w grze. • Programy lojalnościowe i retencja graczy. 12. Podstawy analityki marketingowej i tworzenie planu marketingowego: • Kluczowe wskaźniki efektywności (KPIs) w marketingu gier. • Narzędzia do analizy danych marketingowych. • Struktura i elementy planu marketingowego dla gry. • Budżetowanie działań marketingowych.		
			K_U01, K_U02 K_U16, K_U20, K_U23	K_K01, K_K02, K_K05, K_K06	Laboratoria: 1. Analiza kampanii marketingowych istniejących gier: • Identyfikacja grupy docelowej, USP, kanałów komunikacji i kluczowych przekazów w wybranych kampaniach. 2. Tworzenie persony gracza i definiowanie USP: • Praktyczne ćwiczenie tworzenia person dla hipotetycznej gry. • Określanie unikalnych cech gry. 3. Warsztat tworzenia materiałów prasowych: • Przygotowanie szkicu notatki prasowej i elementów press kitu dla gry. 4. Planowanie strategii w mediach społecznościowych: • Opracowanie przykładowego kalendarza treści dla wybranej platformy social media. • Tworzenie przykładowych postów promocyjnych. 5. Projektowanie strony produktu w sklepie cyfrowym: • Analiza i projektowanie kluczowych elementów strony gry na Steam lub w Google Play (opis, tagi, grafiki, trailer). 6. Symulacja wyboru influencera i propozycja współpracy: • Wyszukiwanie i analiza potencjalnych influencerów dla danego typu gry. • Przygotowanie szkicu propozycji współpracy. 7. Podstawy tworzenia trailera/materiału wideo: • Analiza dobrych praktyk, storyboard dla krótkiego teasera lub gameplay video. 8. Analiza danych i optymalizacja kampanii (symulacja): • Praca na przykładowych danych, identyfikacja wniosków i propozycji usprawnień dla hipotetycznej kampanii. 9. Opracowanie zarysu planu marketingowego dla małej gry:		Projekt marketingowy

					• Zdefiniowanie celów, grupy docelowej, budżetu (hipotetycznego), kluczowych działań i harmonogramu dla projektu gry (np. stworzonego na innych przedmiotach). • Prezentacja projektu planu marketingowego.		
27.	Zaawansowane programowanie w C++/C#	K_W02, K_W10, K_W12, K_W14,		K_K01	Wykłady: 1. Zaawansowane programowanie AI dla postaci niezależnych (NPC) w silnikach: • Maszyny Stanów Skończonych (FSM): Implementacja hierarchicznych FSM w C#/C++ dla Unity/Unreal, zarządzanie złożonymi zachowaniami, integracja z systemem animacji. • Drzewa Behawioralne (Behavior Trees - BT): Projektowanie i implementacja BT w Unity (np. z użyciem assetów lub własnych rozwiązań) i Unreal Engine (natywne BT). Tworzenie własnych węzłów (Tasks, Services, Decorators). Debugowanie i optymalizacja BT. • Algorytmy wyszukiwania drogi: Algorytm Dijkstry, Algorytm A* (A-star) i inne – implementacja i zastosowanie w grach. • Systemy Nawigacji (Navigation Systems): Zaawansowane wykorzystanie NavMesh, dynamiczne omijanie przeszkód, pathfinding dla grup agentów, modyfikowanie siatki nawigacyjnej w czasie rzeczywistym (np. NavMesh Modifiers). 2. Programowanie proceduralne i generowanie treści w silnikach: • Generowanie poziomów i geometrii: Tworzenie skryptów generujących meshe, rozmieszczających obiekty, tworzących układy pomieszczeń. • Zastosowanie szumu proceduralnego (Perlin, Simplex, FBM): Generowanie terenu, tekstur proceduralnych, efektów specjalnych w silniku. • L-Systemy w praktyce: Implementacja generowania roślinności lub innych struktur fraktalnych w Unity/Unreal. • Integracja systemów proceduralnych z edytorem silnika (np. tworzenie niestandardowych narzędzi edytorskich). 3. Zaawansowane systemy animacji i kontrola postaci: • Proceduralna kontrola animacji: Kinematyka odwrotna (IK) dla pozycjonowania kończyn, celowania, interakcji z otoczeniem – implementacja i konfiguracja w Unity/Unreal. • Złożone kontrolery animacji: Zaawansowane State Machines, Blend Trees (1D, 2D, Direct), warstwy animacji, maski, synchronizacja animacji dla wielu postaci.	EGZ	Egzamin - test

					• Systemy fizycznej animacji (Physical Animation / Ragdolls) – podstawy implementacji. • Dynamiczne dostosowywanie animacji do terenu i sytuacji (np. adaptive foot placement). 4. Zaawansowana fizyka i interakcje w silnikach: • Symulacja złożonych obiektów fizycznych: Implementacja pojazdów z realistycznym zawieszeniem i modelem jazdy. • Niestandardowe interakcje fizyczne: Tworzenie niestandardowych sił, efektów fizycznych, systemów zniszczeń. • Praca z zaawansowanymi połączeniami (Configurable Joints w Unity, Physics Constraints w Unreal). • Optymalizacja scen z dużą liczbą obiektów fizycznych (np. techniki grupowania, upraszczania kolizji). 5. Zaawansowane techniki programowania w C# dla Unity: • Delegaty, zdarzenia i wyrażenia lambda w kontekście systemów gry (np. systemy zdarzeń, komunikacja między komponentami). • Programowanie asynchroniczne (async/await) do obsługi operacji długotrwałych (np. ładowanie zasobów). • Zaawansowane wykorzystanie ScriptableObjects do architektury systemów i zarządzania danymi. • Tworzenie niestandardowych edytorów i narzędzi (Editor scripting) w celu usprawnienia pracy zespołu. 6. Zaawansowane techniki programowania w C++ dla Unreal Engine: • Zaawansowane wykorzystanie szablonów (templates) w kontekście systemów gry. • Inteligentne wskaźniki i zarządzanie pamięcią w złożonych systemach. • System refleksji (Reflection System) i jego wykorzystanie (np. do tworzenia systemów serializacji, narzędzi). • Modułowość i tworzenie własnych pluginów. • Interfejsy C++ i ich rola w tworzeniu elastycznych systemów. 7. Architektura kodu i wzorce projektowe w dużych projektach gier: • Skalowalne systemy oparte na zdarzeniach (Event-Driven Architecture). • Wzorce Entity-Component-System (ECS) – głębsze zrozumienie i implementacja (np. DOTS w Unity – wprowadzenie koncepcyjne).		

					• Zarządzanie zależnościami (Dependency Injection – wprowadzenie do Zenject). • Utrzymanie czystości kodu i refaktoryzacja w kontekście silników.		
			K_U01, K_U06, K_U10, K_U11, K_U12, K_U13, K_U19, K_U22, K_U25	K_K01	Laboratoria: 1. Implementacja zaawansowanego agenta AI z wykorzystaniem Drzew Behawioralnych: • Tworzenie złożonych zachowań (patrolowanie, poszukiwanie, walka, ucieczka) w Unreal Engine lub z assetem w Unity. 2. Implementacja systemu pathfindingu A* i integracja z NavMesh silnika: • Porównanie wyników, obsługa dynamicznych przeszkód. 3. Proceduralne generowanie fragmentu poziomu lub obiektu w silniku: • Np. generowanie labiryntu, rozmieszczanie obiektów według reguł proceduralnych. 4. Implementacja systemu IK dla postaci: • Np. celowanie bronią, dopasowanie stóp do nierównego terenu. 5. Tworzenie złożonego kontrolera animacji dla postaci: • Wykorzystanie blend spaces, warstw, masek do płynnych przejść i złożonych zachowań. 6. Implementacja zaawansowanej mechaniki fizycznej: • Np. prosty system fizyki pojazdu, system zniszczeń oparty na fizyce. 7. Stworzenie niestandardowego narzędzia edytorskiego w Unity lub Unreal: • Np. edytor do rozmieszczania obiektów, konfigurator postaci. 8. Refaktoryzacja istniejącego kodu z zastosowaniem wzorców projektowych: • Poprawa struktury i czytelności przykładowego systemu gry. 9. Projekt – implementacja zaawansowanego systemu gry: • Np. system walki dla postaci, system jazdy pojazdem, zaawansowany system AI dla grupy przeciwników, generator proceduralny złożonych struktur, system fizyki dla specyficznego obiektu. Projekt realizowany w wybranym silniku.		Zadania laboratoryjne, dokumentacja projektowa

28.	Projekt zespołowy	K_W02, K_W10, K_W11, K_W13, K_W14, K_W16, K_W19, K_W20		K_K01	Wykłady: 1. Wprowadzenie do projektu zespołowego: • Cele przedmiotu, oczekiwania, harmonogram. • Zasady formowania zespołów interdyscyplinarnych (programiści + graficy/level designerzy). • Omówienie ról w zespole projektowym (np. Project Lead/Manager, Programista Główny, Grafik Główny, Level Designer, UI/UX Designer, Tester – role mogą być płynne i łączone). 2. Metodyki zarządzania projektami w gamedevie (wprowadzenie praktyczne): • Agile (Scrum/Kanban) w kontekście projektu studenckiego. • Planowanie sprintów, codzienne spotkania (stand-upy – nawet w uproszczonej formie), retrospektywy. • Narzędzia do zarządzania projektami (np. Trello, Jira, Asana – wybór jednego, prostego narzędzia dla wszystkich zespołów). 3. Definiowanie zakresu projektu (Scope): • Określanie realistycznego zakresu gry możliwego do zrealizowania w ramach semestru. • MVP (Minimum Viable Product) – koncepcja i jej zastosowanie. • Ryzyka projektowe i sposoby ich minimalizacji. 4. Dokumentacja projektowa (GDD – Game Design Document): • Kluczowe elementy GDD dla projektu zespołowego (koncepcja gry, mechaniki, docelowa platforma, styl graficzny, zarys fabuły/poziomów, wymagania techniczne). • Znaczenie aktualizacji dokumentacji w trakcie trwania projektu. 5. Narzędzia kontroli wersji (np. Git): • Podstawy pracy z Git w zespole (repozytoria, branche, commity, merge). • Dobre praktyki współpracy przy użyciu Git. 6. Komunikacja i rozwiązywanie konfliktów w zespole: • Efektywna komunikacja w zespole projektowym. • Techniki rozwiązywania problemów i podejmowania decyzji. 7. Prezentacja postępów i demo gry: • Przygotowanie do prezentacji "milestone'ów" i finalnego dema. • Jak efektywnie prezentować swoją pracę.	ZAO	Test, prezentacja

			K_U01, K_U02, K_U03, K_U04, K_U06, K_U22, K_U25, K_U26	K_K01, K_K02, K_K03, K_K04, K_K05	Laboratoria / lub Praca zespołowa: 1. Faza koncepcyjna i formowanie zespołów: • Burza mózgów, generowanie pomysłów na gry. • Formowanie zespołów, wybór liderów/koordynatorów. • Wstępne zdefiniowanie koncepcji gry i jej zakresu. 2. Tworzenie dokumentu GDD (iteracyjnie): • Opracowanie pierwszej wersji GDD. • Konsultacje z prowadzącym, iteracyjne doskonalenie dokumentu. 3. Planowanie i organizacja pracy: • Podział zadań w zespole. • Ustalenie harmonogramu prac, definicja sprintów/milestone'ów. • Konfiguracja narzędzi do zarządzania projektem i kontroli wersji. 4. Faza prototypowania (Proof of Concept / Vertical Slice): • Implementacja kluczowych mechanik gry. • Stworzenie prototypu graficznego i/lub fragmentu poziomu. • Testowanie podstawowych założeń gry. 5. Rozwój gry – Iteracja 1 (np. pierwszy grywalny build): • Implementacja kolejnych funkcjonalności zgodnie z GDD i planem. • Tworzenie zasobów graficznych, projektowanie poziomów. • Regularne spotkania zespołowe, rozwiązywanie bieżących problemów. • Prezentacja postępów (Milestone 1). 6. Rozwój gry – Iteracja 2 (np. rozbudowa i polerowanie): • Dodawanie treści (poziomy, postacie, przeciwnicy, zadania). • Integracja grafiki, dźwięku, UI. • Testowanie wewnętrzne w zespole, zbieranie feedbacku. • Prezentacja postępów (Milestone 2). 7. Testowanie i optymalizacja (podstawowy zakres): • Przeprowadzanie testów funkcjonalnych i playtestów. • Identyfikacja i naprawa błędów. • Podstawowa optymalizacja gry. 8. Przygotowanie do finalnej prezentacji: • Finalizacja grywalnego dema. • Przygotowanie materiałów prezentacyjnych (np. krótki gameplay video, slajdy).		Dokumentacja projektowa, sprawozdanie

					9. Finalna prezentacja projektu zespołowego: • Prezentacja gry, omówienie procesu deweloperskiego, napotkanych wyzwań i osiągnięć. • Sesja Q&A. 10. Dokumentacja końcowa: • Złożenie finalnej wersji GDD, kodu źródłowego, zasobów i grywalnego builda. • Krótkie sprawozdanie z realizacji projektu przez zespół.		
29.	Gry sieciowe i podstawy programowania sieciowego	K_W07, K_W10, K_W13, K_W14		K_K01	Wykłady: 1. Wprowadzenie do programowania sieciowego: • Podstawowe pojęcia: klient, serwer, host, protokoły (TCP/IP, UDP), porty, gniazda (sockets). • Modele komunikacji sieciowej: klient-serwer, peer-to-peer (P2P). • Różnice między TCP a UDP i ich zastosowania w grach. • Podstawy adresacji IP i DNS. 2. Architektury gier sieciowych: • Autorytatywny serwer vs. nieautorytatywny serwer. • Topologie sieciowe w grach (gwiazda, siatka). • Wyzwania w tworzeniu gier sieciowych: opóźnienia (latency), synchronizacja, przepustowość (bandwidth), utrata pakietów. 3. Synchronizacja stanu gry: • Konieczność synchronizacji i podstawowe problemy. • Techniki synchronizacji: dead reckoning (ekstrapolacja), client-side prediction (predykcja po stronie klienta), server reconciliation (uzgadnianie po stronie serwera). • Synchronizacja obiektów, atrybutów i zdarzeń. • Delta compression i optymalizacja przesyłanych danych. 4. Komunikacja w grach sieciowych: • Zdalne wywołania procedur (RPCs) – koncepcja i zastosowanie. • Niezawodne i nieniezawodne przesyłanie wiadomości. • Serializacja danych do transmisji sieciowej. 5. Programowanie sieciowe w silniku Unity: • Przegląd dostępnych rozwiązań (np. Netcode for GameObjects, Mirror, Photon – omówienie koncepcji). • Podstawy konfiguracji projektu sieciowego. • Komponenty sieciowe: NetworkObject, NetworkBehaviour. • Synchronizacja transformacji (NetworkTransform) i zmiennych (NetworkVariable/SyncVar).	ZAO	Kolokwium - test

					• Implementacja RPC (ServerRpc, ClientRpc). • Zarządzanie obiektami sieciowymi (spawnowanie, despawnowanie). 6. Programowanie sieciowe w silniku Unreal Engine: • Architektura sieciowa Unreal Engine (Replication). • Aktorzy i komponenty w kontekście sieciowym (Relevancy, NetCullDistanceSquared). • Replikacja zmiennych (Replicated, RepNotify). • Implementacja RPC (Client, Server, NetMulticast). • Rola serwera i klienta w modelu replikacji. • Replikacja ruchu postaci (CharacterMovementComponent). 7. Zarządzanie sesją i graczami: • Podstawy matchmakingu i systemów lobby. • Obsługa dołączania i odłączania graczy. • Utrzymywanie stanu sesji gry. 8. Bezpieczeństwo w grach sieciowych (wprowadzenie): • Najczęstsze zagrożenia: oszustwa (cheating), ataki DoS/DDoS. • Podstawowe techniki zabezpieczania komunikacji i logiki gry. • Walidacja danych wejściowych po stronie serwera. 9. Testowanie i debugowanie gier sieciowych: • Narzędzia do symulacji warunków sieciowych (opóźnienia, utrata pakietów). • Techniki debugowania logiki sieciowej. • Testowanie różnych scenariuszy połączeń. 10. Optymalizacja wydajności gier sieciowych: • Redukcja ilości przesyłanych danych. • Optymalizacja częstotliwości aktualizacji. • Interest management – zarządzanie zainteresowaniem (wysyłanie tylko istotnych danych).		
			K_U01, K_U10, K_U11, K_U19, K_U22, K_U25	K_K01, K_K04	Laboratoria: 1. Podstawy komunikacji socketowej (opcjonalnie, jako wprowadzenie): • Implementacja prostego czatu klient-serwer (TCP i/lub UDP) w C# lub C++. 2. Konfiguracja projektu sieciowego w Unity/Unreal Engine: • Utworzenie projektu, podstawowe ustawienia sieciowe. • Implementacja prostego połączenia klient-serwer. 3. Synchronizacja ruchu postaci w Unity/Unreal Engine:		Projekt sieciowy

					• Implementacja podstawowej synchronizacji transformacji dla obiektów graczy. • Testowanie płynności ruchu przy różnych warunkach sieciowych (symulowanych). 4. Implementacja zdalnych wywołań procedur (RPC): • Przesyłanie zdarzeń i akcji między klientami a serwerem (np. strzał, użycie przedmiotu). 5. Synchronizacja stanu obiektów i zmiennych gry: • Implementacja synchronizacji atrybutów gracza (np. punkty życia, amunicja) lub stanu obiektów w grze. 6. Implementacja podstawowej logiki gry sieciowej: • Stworzenie prostej mechaniki gry wieloosobowej (np. zbieranie punktów, prosta interakcja między graczami). 7. Zaawansowane techniki synchronizacji (wprowadzenie praktyczne): • Implementacja uproszczonej predykcji po stronie klienta dla ruchu postaci. • Podstawy uzgadniania stanu po stronie serwera. 8. Zarządzanie graczami i sesją: • Implementacja prostego systemu listy graczy. • Obsługa dołączania/odłączania graczy w trakcie gry. 9. Testowanie i debugowanie aplikacji sieciowej: • Praktyczne wykorzystanie narzędzi do debugowania i profilowania kodu sieciowego. • Identyfikacja i rozwiązywanie typowych problemów sieciowych. 10. Projekt grupowy/indywidualny – Prosta gra sieciowa: • Zaprojektowanie i zaimplementowanie małej, grywalnej gry wieloosobowej (np. gra typu "tag", prosty shooter, gra kooperacyjna) z wykorzystaniem poznanych technik w wybranym silniku. Prezentacja projektu.		
30.	Programowanie w Open GL i Shaderów 3D	K_W01, K_W03, K_W06, K_W14, K_W16		K_K01	Wykłady: 1. Wprowadzenie do grafiki 3D i API graficznych: • Podstawowe pojęcia grafiki 3D: wierzchołki, prymitywy, transformacje, macierze, wektory, przestrzeń świata, widoku, projekcji. • Rola API graficznych (OpenGL, DirectX, Vulkan, Metal). • Architektura potoku renderowania (Graphics Pipeline) – stało- i programowalnofunkcyjny. 2. Podstawy OpenGL: • Historia i wersje OpenGL. Kontekst renderowania, okno.	ZAO	Kolokwium pisemne

					• Ładowanie funkcji OpenGL (np. przy użyciu bibliotek typu GLEW, GLAD). • Vertex Buffer Objects (VBO), Vertex Array Objects (VAO), Element Buffer Objects (EBO). • Rysowanie prymitywów (trójkąty, linie, punkty). 3. Transformacje w OpenGL: • Macierze transformacji: translacja, rotacja, skalowanie. • Przestrzenie współrzędnych: lokalna, świata, widoku (kamery), odcinania (clip space), znormalizowanych współrzędnych urządzenia (NDC), ekranu. • Macierz modelu, widoku i projekcji (perspektywiczna, ortogonalna). • Przekazywanie macierzy do shaderów (uniformy). 4. Wprowadzenie do języka shaderów GLSL (OpenGL Shading Language): • Rola shaderów w programowalnym potoku renderowania. • Typy shaderów: Vertex Shader, Fragment Shader (Pixel Shader). • Podstawy składni GLSL: typy danych (vec, mat, sampler), zmienne (in, out, uniform), funkcje wbudowane. • Kompilacja i linkowanie programów shaderowych. 5. Vertex Shader – programowanie: • Przetwarzanie atrybutów wierzchołków (pozycja, kolor, współrzędne tekstury). • Transformacje wierzchołków. • Przekazywanie danych do Fragment Shadera (interpolacja). 6. Fragment Shader – programowanie: • Obliczanie koloru finalnego piksela. • Podstawowe operacje na kolorach. • Proste efekty (np. zmiana koloru, gradient). 7. Teksturowanie w OpenGL: • Pojęcie tekstury, współrzędne tekstury (UV). • Ładowanie plików graficznych jako tekstur (np. przy użyciu bibliotek typu stb_image). • Generowanie i bindowanie obiektów tekstur. • Parametry tekstur: filtrowanie (najbliższego sąsiada, liniowe), wrapping (repeat, clamp). • Wykorzystanie samplerów 2D w shaderach. 8. Oświetlenie w grafice 3D – modele oświetlenia: • Podstawowe typy świateł: ambient, diffuse, specular. • Model oświetlenia Phonga i Blinna-Phonga.		

					• Wektory normalne i ich rola w oświetleniu. • Implementacja oświetlenia w shaderach (Vertex i Fragment Shader). 9. Zaawansowane techniki shaderowe (wprowadzenie): • Normal mapping (mapowanie normalnych) dla dodania szczegółowości powierzchni. • Specular mapping (mapowanie odbić lustrzanych). • Proste efekty post-processingu (np. skala szarości, sepia, rozmycie – koncepcja framebuffera). • Podstawy przezroczystości i blendingu. 10. Integracja OpenGL z bibliotekami pomocniczymi: • Biblioteki do tworzenia okna i obsługi wejścia (np. GLFW, SDL). • Biblioteki matematyczne (np. GLM). 11. Debugowanie aplikacji OpenGL i shaderów: • Narzędzia do debugowania (np. RenderDoc – wprowadzenie, glGetError). • Typowe błędy i problemy w programowaniu OpenGL.		
			K_U01, K_U07, K_U10, K_U12, K_U15	K_K01	Laboratoria: 1. Konfiguracja środowiska i pierwsze okno OpenGL: • Instalacja niezbędnych bibliotek (GLEW/GLAD, GLFW, GLM). • Stworzenie pierwszego okna OpenGL i pętli renderowania. 2. Rysowanie pierwszego trójkąta i podstawy VBO/VAO: • Definiowanie wierzchołków, tworzenie VBO i VAO. • Napisanie prostego Vertex i Fragment Shadera (np. stały kolor). 3. Transformacje 2D/3D i przekazywanie uniformów: • Implementacja macierzy modelu, widoku, projekcji. • Animowanie obiektu poprzez zmianę macierzy transformacji. 4. Kolory i interpolacja w shaderach: • Przekazywanie kolorów jako atrybutów wierzchołków. • Obserwacja interpolacji kolorów w Fragment Shaderze. 5. Teksturowanie obiektów: • Ładowanie tekstury z pliku. • Aplikowanie tekstury na prostym obiekcie (np. kwadrat, sześcian). 6. Implementacja podstawowego oświetlenia (model Phonga): • Definiowanie właściwości materiału i światła.		Zadana laboratoryjne, sprawozdanie z ćwiczeń

					• Implementacja obliczeń oświetlenia ambient, diffuse i specular w shaderach. • Wykorzystanie wektorów normalnych. 7. Normal Mapping: • Ładowanie mapy normalnych. • Modyfikacja shadera oświetlenia w celu wykorzystania normal mappingu. 8. Proste efekty post-processingu (np. skala szarości): • Renderowanie sceny do framebuffera. • Aplikowanie efektu na teksturze z framebuffera w drugim przebiegu renderowania. 9. Ładowanie prostych modeli 3D (opcjonalnie, np. format .obj): • Wykorzystanie biblioteki do ładowania modeli (np. assimp – wprowadzenie). • Renderowanie załadowanego modelu z wykorzystaniem napisanych shaderów. 10. Projekt końcowy – mała scena 3D z własnymi shaderami: • Student tworzy prostą scenę 3D (np. kilka obiektów, oświetlenie, teksturowanie) z wykorzystaniem napisanych przez siebie shaderów, demonstrując zrozumienie poznanych technik. Prezentacja projektu i kodu.		
31.	Testowanie i optymalizacja gier	K_W10, K_W14		K_K01	Wykłady: 1. Wprowadzenie do testowania gier komputerowych: • Rola i znaczenie testowania w cyklu produkcyjnym gry. • Rodzaje testów: funkcjonalne, wydajnościowe, użyteczności (playtesting), kompatybilności, regresji, alfa, beta, smoke testy. • Metodologie testowania (np. testowanie eksploracyjne, oparte na przypadkach testowych). • Proces zgłaszania i zarządzania błędami (bug tracking). 2. Wprowadzenie do optymalizacji gier: • Dlaczego optymalizacja jest kluczowa? (wydajność, płynność, doświadczenie gracza, wymagania platform). • Złote zasady optymalizacji (np. "nie optymalizuj przedwcześnie", profiluj zanim zoptymalizujesz). • Identyfikacja wąskich gardeł (bottlenecks): CPU-bound vs. GPU-bound. • Metryki wydajności: FPS, czas renderowania klatki, użycie pamięci, zużycie procesora. 3. Profilowanie gier w silniku Unity: • Narzędzie Unity Profiler: przegląd okien (CPU Usage, GPU Usage, Rendering, Memory, Audio, Physics).	EGZ	Egzamin pisemny

					• Analiza danych profilera: identyfikacja kosztownych funkcji, skryptów, zasobów. • Deep Profiling. • Frame Debugger do analizy kolejności i kosztu operacji renderowania. • Memory Profiler (Package) do szczegółowej analizy zużycia pamięci. 4. Profilowanie gier w silniku Unreal Engine: • Narzędzia Unreal Insights: śledzenie zdarzeń, analiza wydajności CPU i GPU, pamięci, sieci. • Statystyki w czasie rzeczywistym (Stat Unit, Stat FPS, Stat GPU, etc.). • Frontend Session (profilowanie CPU). • GPU Visualizer do analizy wydajności GPU. • Memory Insights (część Unreal Insights). 5. Optymalizacja kodu i skryptów (CPU-bound): • Techniki optymalizacji kodu C#: unikanie kosztownych operacji w pętlach (np. GetComponent, FindObjectOfType w Update), caching referencji. • Zarządzanie pamięcią w C#: Garbage Collector (GC), minimalizacja alokacji pamięci, stosowanie struktur zamiast klas tam, gdzie to możliwe, Object Pooling. • Wykorzystanie Coroutines i zadań asynchronicznych (Async/Await) do rozkładania obciążenia. • Optymalizacja logiki gry, algorytmów (np. pathfinding, AI). • Specyfika optymalizacji kodu C++ w Unreal Engine: zarządzanie pamięcią, optymalizacja pętli, wykorzystanie natywnych struktur danych. • Optymalizacja Blueprintów w Unreal Engine (kiedy konwertować na C++). 6. Optymalizacja grafiki (GPU-bound): • Modele 3D: Redukcja liczby wielokątów (polycount), poziomy szczegółowości (LODs). • Tekstury: Optymalne rozmiary, formaty kompresji (DXT, ASTC, ETC), mipmapy, atlasy tekstur. • Materiały i Shadery: Upraszczanie shaderów, unikanie kosztownych operacji, analiza złożoności shaderów. • Oświetlenie i Cienie: Optymalizacja źródeł światła (statyczne vs. dynamiczne), mapy świetlne (lightmaps), jakość i zasięg cieni, techniki cieniowania (np. Cascaded Shadow Maps).		

					• Overdraw: Identyfikacja i redukcja (np. przez sortowanie przezroczystych obiektów, optymalizacja UI). • Culling: Frustum Culling, Occlusion Culling (konfiguracja i wykorzystanie w Unity i Unreal Engine). • Batching/Instancing: Dynamic Batching, Static Batching (Unity), GPU Instancing (Unity/Unreal) w celu redukcji liczby wywołań rysowania (draw calls). 7. Optymalizacja fizyki: • Upraszczanie koliderów (primitive colliders vs. mesh colliders). • Optymalizacja ustawień silnika fizycznego (np. częstotliwość aktualizacji, liczba iteracji solvera). • Warstwy kolizji (Collision Layers) do ograniczenia liczby sprawdzanych interakcji. 8. Optymalizacja UI: • Optymalizacja renderingu elementów UI (np. Canvas w Unity – podział na wiele Canvasów, unikanie częstych przebudów). • Redukcja overdraw w interfejsach. • Użycie atlasów dla grafik UI. 9. Zarządzanie zasobami i pamięcią: • Techniki ładowania zasobów: ładowanie synchroniczne vs. asynchroniczne. • Unity: AssetBundles, Addressables. • Unreal Engine: Pakowanie i Chunking, system montowania paków. • Strategie ładowania i zwalniania zasobów w celu uniknięcia skoków wydajności i przekroczenia limitów pamięci. 10. Testowanie wydajności i regresji: • Definiowanie scenariuszy testów wydajnościowych. • Automatyzacja testów wydajnościowych (wprowadzenie). • Monitorowanie wydajności na różnych platformach docelowych. • Znaczenie testów regresji w kontekście optymalizacji. 11. Specyfika optymalizacji dla platform mobilnych i VR/AR: • Ograniczenia sprzętowe (CPU, GPU, pamięć, bateria). • Szczególne wyzwania i techniki optymalizacji (np. redukcja draw calls, optymalizacja shaderów pod kątem urządzeń mobilnych, unikanie choroby symulatorowej w VR poprzez stabilny FPS).		

			K_U01, K_U03, K_U07, K_U10, K_U22	K_K01	Laboratoria: 1. Wprowadzenie do narzędzi profilujących w Unity: • Konfiguracja i uruchomienie Unity Profiler na przykładowym projekcie. • Analiza podstawowych metryk, identyfikacja problematycznych obszarów. 2. Wprowadzenie do narzędzi profilujących w Unreal Engine: • Konfiguracja i uruchomienie Unreal Insights lub statystyk czasu rzeczywistego na przykładowym projekcie. • Analiza podstawowych metryk, identyfikacja problematycznych obszarów. 3. Praktyczna optymalizacja skryptów w Unity: • Identyfikacja i refaktoryzacja nieoptymalnego kodu C# (np. nadużywanie GetComponent w Update, częste alokacje). • Implementacja Object Poolingu dla często tworzonych i niszczonych obiektów. 4. Praktyczna optymalizacja kodu/Blueprintów w Unreal Engine: • Identyfikacja i refaktoryzacja nieoptymalnych fragmentów kodu C++ lub Blueprintów. • Testowanie wpływu konwersji Blueprintów na C++ (jeśli dotyczy). 5. Optymalizacja grafiki – Modele i Tekstury: • Tworzenie i konfiguracja LODs dla modeli w Unity/Unreal Engine. • Analiza i optymalizacja rozmiarów i formatów tekstur w przykładowej scenie. 6. Optymalizacja grafiki – Oświetlenie, Cienie i Culling: • Konfiguracja lightmappingu (baked lighting) w Unity/Unreal Engine. • Testowanie różnych ustawień cieni i ich wpływu na wydajność. • Praktyczne zastosowanie Occlusion Culling. 7. Optymalizacja grafiki – Draw Calls: • Analiza liczby draw calls w scenie. • Praktyczne zastosowanie Static Batching i GPU Instancing w Unity. • Praktyczne zastosowanie Instanced Static Meshes w Unreal Engine. 8. Optymalizacja fizyki i UI: • Analiza i optymalizacja koliderów w przykładowej scenie. • Identyfikacja i optymalizacja problemów z wydajnością UI (np. przebudowa Canvas w Unity).		Zadanie laboratoryjne, sprawozdanie.

					9. Zarządzanie pamięcią i zasobami: • Analiza zużycia pamięci za pomocą Memory Profiler (Unity) lub Memory Insights (Unreal). • Praktyczne ćwiczenia z ładowaniem zasobów (np. proste użycie Addressables w Unity). 10. Projekt optymalizacyjny: • Student otrzymuje projekt gry (lub jego fragment) z celowo wprowadzonymi problemami wydajnościowymi. • Zadaniem jest zdiagnozowanie problemów za pomocą narzędzi profilujących, zaproponowanie i zaimplementowanie optymalizacji w co najmniej dwóch obszarach (np. kod i grafika), a następnie udokumentowanie i zaprezentowanie wyników (poprawa FPS, redukcja zużycia pamięci itp.).		
32.	Techniki i narzędzia grafiki 2D	K_W02 K_W16 K_W17 K_W14 K_W15 K_W20		K_K01	Wykłady: 1. Praca z grafiką rastrową w stylistyce pixel art. w programie Aseprite/Libresprite. 2. Najważniejsze zasady pracy z arkuszami spritów (Sprite sheet) dobór odpowiedniej rozdzielczości. Tworzenie i praca z paletami kolorów. Narzędzia Aseprite/Libre sprite. 3. Podstawowe zasady tworzenia animacji poklatkowej. Narzędzia do pracy z animacją poklatkową w Aseprite/Libresprite. 4. Zasady przygotowania elementów otoczenia (enviroment) na gridzie tak aby można było zbudować cały poziom bezpośrednio w silniku. 5. Eksport bitmap do silnika Unity. Import i Sprite Editor w Unity. Dostosowanie edytora, praca na gridzie, snapping, tworzenie poziomu z uprzednio przygotowanych modułów 2d. 6. Zastosowanie Normal Map w pikselowych grafikach rastrowych. 7. Praca z grafiką rastrową - digital painting z wykorzystaniem tabletu graficznego w Adobe Photoshop. Wykorzystanie narzędzi malarskich programu. Zasady pracy nad kompozycją od szkicu do szczegółu.	EGZ	Egzamin pisemny
			K_U01 K_U02 K_U06 K_U15 K_U17 K_U18	K_K01 K_K04 K_K06	Laboratoria: 1. Praca z grafiką rastrową w stylistyce pixel art. w programie Aseprite/Libresprite: • stworzenie Sprite Sheet atlasu środowiska w stylistyce pixel art. Wymiar podstawowego elementu 32/32 pixele.		2 Zadania laboratoryjne - Przygotowanie zestawu spritów elementów do budowy

			K_U14 K_U21		całościowy grid może mieć wymiar 128/128 pixeli. Przygotowanie palety kolorów. • praca nad postaciami wykorzystywanymi w grze. Postać głównego bohatera animacja przygotowana na 6-8 klatek. Przygotowanie animacji Idle, Walk, Attack, Death • eksport bitmap do Unity. Import i przygotowanie spritów oraz animacji w Unity. Ustawienie z przygotowanych assetów, przykładowego poziomu w Unity 2. Praca z grafiką rastrową - digital painting z wykorzystaniem tabletu graficznego w Adobe Photoshop: • przygotowanie grafiki koncepcyjnej w zadanej rozdzielczości np 1920/1080 px. • praktyczna praca z tabletem graficznym.		środowiska. Przygotowanie grafiki rastrowej, w konwencji digital painting
33.	Podstawy fotogrametrii	K_W01 K_W03 K_W14 K_W16 K_W17 K_W18 K_W20		K_K01	Wykłady: 1. Definicja fotogrametrii i przykłady jej zastosowania w grafice 3D do gier/filmów, geodezji, archeologii, konserwacji czy dokumentacji przemysłowej. 2. Sprzęt fotograficzny i optyczny. Podstawy fotografii cyfrowej. Zalecane typy aparatów (lustrzanki, bezlusterkowce). Zalecane obiektywy (zniekształcenia, optyka stałoogniskowa vs. zmiennoogniskowa). Statywy, rotatory, oświetlenie sztuczne (lampy, softboxy, filtry polaryzacyjne). Ustawienia aparatu: ISO, przysłona, czas, ostrość manualna. 3. Zasady fotografowania pod fotogrametrię Jak fotografować obiekty: ilość zdjęć, pokrycie, kąty, głębia ostrości. Warunki studyjne vs. zewnętrzne (światło naturalne, cienie). 4. Narzędzia do obróbki zdjęć RAW. Darktable, Adobe Lightroom – podstawy wywoływania. Korekcja kolorów, ostrości, kontrastu. Eksportowanie zdjęć do dalszej obróbki fotogrametrycznej. 5. Przykładowe narzędzia do fotogrametrii Agisoft Metashape, Reality Capture, Meshroom (open source). Mobilne aplikacje (RealityScan, Polycam) – zalety i ograniczenia. 6. Tworzenie modelu 3D z fotografii Kroki: alignacja, dense cloud, mesh, teksturowanie. Wymagania sprzętowe, czas przetwarzania. Naprawa modelu 3D i optymalizacja. Detekcja i uzupełnienie dziur, edycja siatki, rekonstrukcja brakujących obszarów. Delighting i baking. Usuwanie cieni z tekstur. Baking tekstury do siatki modelu (Albedo, AO).	ZAO	Test pisemny

					7. Generowanie map materiałów PBR. Mapy: roughness, metalness, normal, AO z tekstury i modelu. Zasady PBR – tekstury fizycznie poprawne. 8. Tworzenie LOD i implementacja modelu do silnika gier Poziomy szczegółowości (LOD0–LOD3). Eksport do silników (Unity, Unreal Engine). Redukcja polycount, optymalizacja UV. 9. Tekstury powtarzalne z użyciem z fotogrametrii. Tworzenie bezszwowych tekstur: skały, ściany, teren. Narzędzia: Materialize, Photoshop/GIMP.		
			K_U01 K_U02 K_U14 K_U15 K_U21 K_U22 K_U23	K_K01 K_K04 K_K05 K_K06	Laboratoria: 1. Warsztat fotograficzny – zasady robienia zdjęć. Praca z różnymi obiektywami, manualne ustawienia. Fotografowanie obiektów pod fotogrametrię (mały obiekt studyjny). 2. Wyjście terenowe. Fotografowanie w plenerze (pomnik, rzeźba, fragment architektury). Praca w grupach, prezentacja fotografii 3. Studio z oświetleniem sztucznym. Tworzenie zestawu zdjęć przy świetle studyjnym. Fotogrametria z użyciem stołu obrotowego (np. skanowanie artefaktu archeologicznego/ produktu). 4. Obróbka RAW w Darktable/Lightroom. Wywołanie zdjęć z sesji studyjnej i terenowej. Eksport zdjęć do JPEG/TIFF 5. Tworzenie modelu 3D w Reality. Capture/Meshroom(open source). Wczytywanie zdjęć, generowanie chmury punktów. Teksturowanie i eksport modelu. 6. Naprawa siatki – MeshLab/Blender. Zamykanie dziur, wygładzanie modelu, siatka UV. 7. Baking i delighting z użyciem Blender usuwanie cieni z tekstur. 8. Generowanie map materiałowych Mapy: normal, roughness, AO, height z tekstury w Materialize. 9. Tworzenie LOD i eksport do silnika. Import modelu do Unity/Unreal, sprawdzenie LOD. Optymalizacja materiałów. 10. Seamless texture – praktyka. Tworzenie pętli tekstury na bazie zdjęć powierzchni (skała, cegła). 11. Projekt indywidualny. Wykonanie pełnego procesu: zdjęcia → model 3D → optymalizacja. Obiekt: dowolny, konsultowany ze studentem. 12. Prezentacja projektów. Ocena modelu, tekstur, poprawności LOD i jakości exportu.		2 zadania laboratoryjne - Model 3d wykonany w technice fotogrametrii. Bezszwowy powtarzalny materiał PBR stworzony z użyciem fotogrametrii

34.	Projektowanie wizualne	K_W02 K_W13 K_W14 K_W15 K_W16 K_W17		K_K01	Wykłady: 1. Historia sztuki - najważniejsze nurty od starożytności do XX wieku 2. Historia sztuki najnowszej XX- XXI wiek ze szczególnym uwzględnieniem filmu, animacji, gier komputerowych. 3. Podstawy budowania kompozycji plastycznej. Kompozycja statyczna, kompozycja dynamiczna. Kompozycja z dominantą kolorystyczną. Rodzaje perspektywy w plastyce. 4. Przegląd stylistyk plastycznych wykorzystywanych w grach komputerowych. Charakterystyka pracy nad określonymi stylistykami graficznymi oraz czas i środki potrzebne na zrealizowanie gry w danej stylistyce. 5. Stylistyka plastyczna w grach komputerowych. 6. Teoria koloru. Kolor i znaczenie. Koło barw. Kolor podstawowy, wtórny, trzeciorzędowy. Zasady budowania palet kolorystycznych. Schematy barw: monochromatyczny, analogiczne, komplementarne. 7. Kontrast. Nasycenie. Walor barwy. 8. Sylwetka i tło. Skalowalność znaku, jego czytelność i minimalne rozmiary. Tło jako przestrzeń negatywna. 9. Typografia. Zasady pracy z tekstem. Formatowanie tekstu. Rodzaje krojów tekstu i ich zastosowanie. 10. Wykorzystanie narzędzi AI jako pomoc w projektowaniu elementów graficznych. Korzystanie z bibliotek i narzędzi internetowych np: Lospec.com	ZAO	Test
			K_U01 K_U02 K_U06 K_U10 K_U13 K_U15 K_U23	K_K02 K_K04 K_K06	Laboratoria: 1. Tworzenie palety kolorystycznej. Używanie wzorników kolorów. Praca z internetowymi narzędziami pomocnymi w doborze kolorów np. Adobe Colors, Coloors.co itp. Paleta powinna nawiązywać do charakteru wymyślonego konceptu gry. 2. Ćwiczenie z zaprojektowania znaku graficznego z wykorzystaniem tylko dwóch kolorów. Z wykorzystaniem negatywnej przestrzeni tła. 3. Praca nad logotypem wymyślonej gry komputerowej. W logotypie powinny zostać zastosowane elementy graficzne i tekst. Projekt przygotowany w wektorach w Adobe Illustrator.		Zadania laboratoryjne - Przygotowanie logotypu wymyślonej gry. Przygotowanie zestawu 3 ikon ikon (np Zdrowie, Ikona ustawień gry czy ikona Karty postaci),

					4. Dobór odpowiedniego kroju fontu do rodzaju i stylu plastycznego gry. 5. Przygotowanie elementów graficznych: kształtów, krzywych spójnych stylistycznie z tekstem. 6. Wykonanie alternatywnych wersji kolorystycznych z zastosowaniem narzędzi AI Adobe Illustrator (Recolor) 7. Ćwiczenie z praktycznym wykorzystaniem narzędzi AI w projektowaniu elementów graficznych: np: projekt logotypu		możliwych do wykorzystania np jako element UI w grze komputerowej.
35.	Modelowanie obiektów 3D	K_W02 K_W16 K_W17 K_W14 K_W20		K_K01	Wykłady: 1. Praca z grafiką 3d do gier w programie Blender. Poly modelling w trybie edit. Zasady pracy z narzędziami Extrude, Bevel, Inset, Loop cut, Knife. Zasady pracy z topologią typu quad. 2. Najważniejsze zasady optymalizacji siatki, łączenie wierzchołków, oddzielanie obiektów modelu 3d. Apply transform. Zaznaczanie określonych grup ścianek (np checker deselect, select similar). Dostosowywanie pivotów obiektów (origin point. Parentowanie obiektów. Praca z modyfikatorem mirror. Modyfikatory Simple Deform, Boolean, Subdivision Surface, Decimate, Remesh, Solidify, Array. 3. Wektory normalne modelu 3d. Weighted normals. Omówienie podstawowych zasad przepalania tekstur. Przepalanie Normal Map, Ambient Occlusion. 4. Wprowadzenie do programu Adobe Substance Painter. Przepalanie tekstur oraz export tekstur na przykładzie materiału typu Trim Sheet. 5. Praca na podstawie planów, rysunków, grafik koncepcyjnych. 6. Tworzenie materiałów z wykorzystaniem tekstur w Blenderze. Przygotowanie map UV. 7. Ręczne malowanie tekstur bezpośrednio na obiekcie 3d - Adobe Substance painter 8. Zasady poprawnego przygotowania obiektów w celu eksportu do zewnętrznych silników growych (np Unity). 9. Wykorzystanie narzędzi AI w tworzeniu modeli 3d, tekstur. 10. Optymalizacja obiektów 3d. Level of detail. Ograniczenie ilości tekstur/materiałów. Zasady pracy z modularnymi assetami, wykorzystanie grida. Przygotowanie tekstury typu Color Gradient Atlas. 11. Texel Density. Teksturowanie z wykorzystaniem materiałów typu overlap oraz non overlap. 12. Import, tworzenie materiałów w silniku Unity lub Unreal	EGZ	Egzamin - test pisemny

			K_U01 K_U02 K_U06 K_U15 K_U23 K_U14 K_U21	K_K04 K_K06	Laboratoria: 1. Praca nad sceną 3d w stylistyce typu low poly, teksturowanej metodą Color Gradient Texture. Eksport gotowej sceny do Sketchfaba lub silnika growego np Unity: • zebranie referencji, przygotowanie sceny do pracy na podstawie rzutów obiektów • praca z obiektami w trybie edycji z wykorzystaniem narzędzi Extrude, Bevel, Inset, Loop cut, Knife z zachowaniem quad topologii siatki obiektu 3d. • zarządzanie origin point obiektów. Dostosowywanie skali. Łączenie wierzchołków obiektu. Łączenie obiektów ze sobą oraz separacja obiektów. • parentowanie obiektów. Praca z modyfikatorem mirror. Modyfikatory Simple Deform, Boolean, Subdivision Surface, Decimate, Remesh, Solidify, Array. • praca z siatką UV modelu. Przygotowanie szwów. Avarge Island scale. Pakowanie UV map. Add on UV Grid of squares. • cieniowanie obiektów - mark sharp. Normalne. Używanie modyfikatora Smooth by angle. Flip normals. Weighted normals. 2. Przygotowanie zestawu modularnych assetów w oparciu o siatkę np. 3m oraz eksport gotowych modeli do silnika growego np. Unity i stworzenie z przygotowanych modułów gotowego poziomu: • przygotowanie zestawu assetów na podstawie referencji. • praca nad teksturą typu Trim. Przygotowanie modelu Hi poly na podstawie Plane no 3/3m • baking Normal Mapy, Ambient Occlusion i przygotowanie tekstur w Adobe Substance Painter • stworzenie zestawu modularnych modeli 3d oteksturowanych przygotowanym w Adobe Substance painter zestawie tekstur typu Trim • przygotowanie modeli do eksportu, eksport do silnika growego np Unity. • przygotowanie Materiałów, prefabów wraz kolizjami w Unity i budowa z przygotowanych Prefabów całego Levelu. • podstawowe opcje oświetlenia, cieniowania, postprocessing w Unity		2 zadania laboratoryjne - Przygotowanie sceny 3d. Przygotowanie zestawu modularnych assetów 3d w oparciu o siatk
36.	Zaawansowane techniki	K_W01 K_W05 K_W16		K_K01	Wykłady: 1. Wprowadzenie do sculptingu. Omówienie najpopularniejszych narzędzi przeznaczonych do pracy z obiektami hi-poly.	ZAO	Test pisemny

	modelowania i animacji postaci	K_W14			2. Zasady sculptingu w Blenderze. Remesh. Multiresolution. Dynatopo. Narzędzia sculptingu Blendera, używanie masek tekstur. Korzystanie z pędzli Blender Kit. Pędzle typu VDM. 3. Zasady pracy z modelami Hi-poly na przykładzie obiektów Hard Surface. 4. Zasady pracy z modelami Hi-poly na przykładzie obiektów Organic. 5. Retopologia. Mask by color ID. 6. Zasady przygotowania obiektów do pracy w Substance Painter wg zasady "Match by mesh name". 7. Praca nad realistycznymi materiałami PBR w Adobe Substance Painter. Pędzle cząsteczkowe Substance Painter. Anchor Points w Substance Painter. 8. Podstawy tworzenia proceduralnych materiałów w Adobe Substance Designer. 9. IK i FK – kinetyka odwrotna i bezpośrednia. Różnice i zastosowania w grach. Zastosowanie inverse kincematics i forward kinematics. 10. Animacja proceduralna i symulacje fizyczne. Kości do symulacji włosów, odzieży, ragdoll, wykorzystanie fizyki w animacji. Integracja z silnikami gier. 11. Wykorzystanie Blendingu animacji w Unity. 12. Obiekty fizyczne, animacja fizyczna. Kości z symulacją fizyki (np. włosy, peleryna). Użycie soft body / jiggle bones / bone physics.		
			K_U01 K_U02 K_U04 K_U10 K_U14 K_U15 K_U22 K_U26	K_K01 K_K04 K_K05	Laboratoria: 1. Przygotowanie obiektu do rzeźbienia w Blenderze z wykorzystaniem narzędzi Remesh, Multiresolution. 2. Sculpting - praca z różnymi rodzajami pędzli, maski, mesh filters. Modyfikacja pędzli, alfy. 3. Hard surface - praca z modelem hi-poly w stylistyce Hard Surface 4. Organic - praca z modelem hi-poly w stylistyce organic 5. Substance Painter - przygotowanie materiałów, eksport do zewnętrznego silnika. 6. Adobe Substance Designer - praca z proceduralnymi materiałami, eksport tekstur do zewnętrznego silnika growego oraz jako Smart Material do Substance Paintera. 7. Animacja z FK i IK. Tworzenie cyklu chodzenia z użyciem kinetyki odwrotnej (IK) i bezpośredniej (FK). Porównanie działania.		3 zadania laboratoryjne

					8. Blending animacji w Unity. Tworzenie płynnych przejść między animacjami walk, idle, attack. Praca z NLA Editor w Blenderze lub Animatorze w Unity. 9. Dodanie dynamicznych kości i fizyki. Dodawanie kości z symulacją fizyki (np. włosy, peleryna). Użycie soft body / jiggle bones / bone physics. 10. Retargeting animacji. Przypisywanie animacji z jednego szkieletu na inny. Dopasowanie motion capture do postaci gracza. 11. Eksport animacji do silnika gry. Eksport modeli i animacji do formatu FBX lub GLTF. Testowanie działania w Unity lub Unreal. 12. Projekt animacji postaci do gry. Indywidualna praca nad zestawem animacji dla jednej postaci: idle, walk, attack, specjalna. 13. Prezentacja i omówienie wykonanych animacji. Feedback od prowadzącego i grupy. Ocena techniczna i artystyczna.		
37.	Fabuła, narracja i udźwiękowienie w grach wideo i wykrywanie incydentów	K_W02 K_W11 K_W16			Wykłady: 1. Podstawowe pojęcia narratologii w grach wideo 2. Struktury opowieści: klasyczne modele (trójaktówka, monomity) w kontekście gier 3. Ludonarracyjny dyskurs – konflikt między mechaniką gry a opowiadaną historią 4. Postacie i charakterystyka bohaterów: kreacja silnych protagonistów i antagonistów 5. Świat gry i budowanie świata (world-building) – spójność fabularna 6. Projektowanie dialogów: zasady pisania dialogów naturalnych i interaktywnych 7. Mechanika rozgałęzionych narracji – projektowanie drzew dialogowych i wybory moralne 8. Pacing narracyjny – tempo opowiadania, momenty kulminacyjne i punkty spowolnienia 9. Lokalizacja i adaptacja tekstów narracyjnych dla różnych rynków 10. Podstawy udźwiękowienia: rola dźwięku (ambient, muzyka, efekty) w budowaniu klimatu 11. Projektowanie efektów dźwiękowych (SFX) – zasady nagrywania, obróbki i implementacji 12. Technologie middleware (FMOD, Wwise) w implementacji dźwięku do silnika gry 13. Synchornizacja animacji postaci z dialogiem – lip-sync i voice acting	ZAO	Kolokwium - test

					14. Narzędzia do szybkiego prototypowania narracji (Twine, Ink, Fungus) 15. Testowanie narracji i audycji dźwięku: playtesting, analiza feedbacku, iteracje		
			K_U17 K_U18	K_K03 K_K04	Laboratoria: 1. Analiza narracyjna istniejącej gry – identyfikacja struktury fabuły i punktów zwrotnych 2. Tworzenie prostego interaktywnego drzewa dialogowego w narzędziu Twine lub Ink 3. Rejestrowanie i montaż krótkiego monologu postaci (voice-over) – podstawy audacity 4. Implementacja prostego systemu dialogowego w Unity 5. Tworzenie krótkiej ścieżki muzycznej w DAW (np. Reaper/FL Studio) i eksport do WAV 6. Projektowanie i import efektów dźwiękowych (SFX) do Unity – rozbijanie plików WAV, ustawienia audio source 7. Podstawy FMOD: konfiguracja projektu, tworzenie zdarzeń dźwiękowych, parametry adaptacyjne 8. Synchronizacja animacji postaci z dialogiem – wykorzystanie LipSync Pro lub mechanizmu w Unity 9. Tworzenie adaptacyjnej ścieżki dźwiękowej w Wwise – warstwowa muzyka w zależności od stanu gry 10. Lokalizacja dialogu: zamiana treści tekstu i dźwięku na wybrany język – przygotowanie plików CSV/XML i import 11. Playtesting narracji: zbieranie feedbacku od grupy testerów, analiza ankiet online (Google Forms) 12. Tworzenie dźwiękowego soundscape'u (ambient) dla fragmentu poziomu – nagrywanie, edycja, implemetacja		3 zadania laboratoryjne
38.	Projektowanie poziomów gier	K_W02, K_W10, K_W11			Wykłady: 1. Wprowadzenie do projektowania poziomów: rola level designu w procesie produkcji gier 2. Podstawowe elementy przestrzeni gry: geografia, siatka, skale i proporcje w środowisku 2D/3D 3. Flow i pacing: kreowanie płynnej ścieżki dla gracza, wykorzystanie linii wzroku i landmarków 4. Funkcjonalność przestrzeni: flow ruchu, skróty, checkpointy i bariery projektowe 5. Narracyjne aspekty poziomu: integracja fabuły z układem przestrzeni, storytelling środowiskowy 6. Modularność i fragmentacja: tworzenie modułów, reużywalność bloków i prefabów	EGZ	Egzamin pisemny

					7. Architektura poziomu: zróżnicowanie stylów architektonicznych, ambient storytelling 8. Balans poziomu trudności: rozkład wrogów, pułapek, zasobów; krzywa trudności 9. Lekcje płynności rozgrywki (flowcharty): wyznaczanie optymalnych ścieżek, punktów decyzyjnych i eksploracji 10. Grafika i czytelność: dobór kolorystyki, oświetlenia i kontrastu w celu ułatwienia orientacji gracza 11. Narzędzia edycyjne: przegląd edytorów (Unity, Unreal Engine, ProBuilder, Tilemap Editor) – podstawy pracy 12. Integracja assetów 2D/3D: pipeline od artysty do silnika – optymalizacja poly count, LOD, tekstury 13. Użycie AI w level designie: generowanie layoutów, automatyczna analiza kolizji, optymalizacja ścieżek NPC 14. Testowanie i iteracje: playtesting, analiza danych telemetrii, feedback loop i poprawki 15. Poziom w grach 2D vs 3D: różnice w podejściu, projekty dwuwymiarowe, side-scrollery i platformówki vs pełne 3D 16. Level design w **VR/AR:** specyfika projektowania przestrzeni immersyjnych – ergonomia, komfort, wejście w interakcje		
			K_U18, K_U26	K_K04, K_K05	Laboratoria: 1. Tworzenie siatki poziomu (2D Tilemap lub Grid w 3D) 2. Modularne bloki (Prefaby/Blueprinty) – zestaw i złożenie podstawowych modułów terenu 3. Kreacja landmarków i elementów orientacyjnych (wybór obiektów, ustawienie, oświetlenie) 4. Integracja assetów 3D i optymalizacja – LOD oraz batching w Unity/Unreal 5. Testowanie balansu poziomu – umieszczenie pułapek i obiektów kolekcjonerskich (skrypt logiki) 6. Rejestracja telemetrii rozgrywki – logowanie pozycji gracza i czasu, eksport do CSV, podstawowa analiza 7. Użycie ProBuilder (Unity) lub Geometry Editing (Unreal) do szybkiego prototypowania layoutu 8. Projektowanie poziomu w VR – konfiguracja teleportacji i podstawowej interakcji z obiektami 9. Analiza telemetrii graczy – generowanie heatmapy w Pythonie i nałożenie na poziom w Unity 10. Iteracyjne wprowadzanie poprawek po playteście – zbieranie feedbacku (Google Forms) i aktualizacja poziomu		Sprawozdanie z laboratorium

39.	Projekt inżynierski	K_KW01, K_W02, K_W08, K_W10, K_W11, K_W13, K_W20	K_U01, K_U03, K_U04, K_U11, K_U13, K_U16, K_U18, K_U26	K_K06	Laboratoria: 1. Omówienie zakresu informatycznego projektu inżynierskiego, jego harmonogramu i formy, oraz ustalenie tematów projektów realizowanych indywidualnie. 2. Wyszukiwanie materiałów źródłowych w informatyce (bazy danych publikacji, zasady cytowania). 3. Ustalenie zasad opracowywania przez studentów rozwiązania problemu praktycznego w informatyce oraz formy jego przedstawienia. 4. Techniki prawidłowego wnioskowania i uogólniania wyników w kontekście projektów inżynierskich. 5. Realizacja projektu inżynierskiego 6. Projekt i implementacja prezentacji dyplomowej	ZAO	Projekt
40.	Seminarium i egzamin dyplomowy	K_W01-K_W20	K_U01, K_U04, K_U06	K_K01, K_K04-K_K06	Seminarium: 1. Omówienie i zaprezentowanie studentom Zarządzenia Rektora dot. zasad dyplomowania w Akademii WSEI 2. Sformułowanie zadań do realizacji przez studentów oraz omówienie warunków zaliczenia seminarium w poszczególnych semestrach, w tym egzaminu dyplomowego; 3. Przydzielenie studentom obszarów zagadnień dyplomowych w poszczególnych semestrach do samodzielnego opracowania; 4. Merytoryczne omówienie zagadnień dyplomowych; 5. Samodzielne opracowanie przez studenta zagadnienia dyplomowego i prezentacji na jego temat; 6. Prezentacja opracowanego zagadnienia dyplomowego przez każdego studenta na forum grupy; 7. Dyskusja na forum grupy, pytania i odpowiedzi; 8. Ocena dokonanego opracowania dla zaliczenia seminarium	EGZ	Prezentacja zagadnienia dyplomowego. Egzamin dyplomowy
41.	Praktyki zawodowe		K_U02, K_U03, K_U04, K_U21, K_U22, K_U25, K_U26	K_K02-K_K04, K_K06	Praktyka: I rok Praktyka zawodowa realizowana w siedzibie Uczelni: 1. Wprowadzenie do praktyk zawodowych i spotkania z praktykami. Praktyka zawodowa realizowana u pracodawcy: 1. Wprowadzenie do praktyk zawodowych; 2. Zapoznanie się ze statusem, strukturą oraz zasadami działania danej jednostki (podmiotu), w którym realizowana jest praktyka zawodowa; 3. Zapoznanie się z normami z zakresu bezpieczeństwa i higieny pracy, przepisami przeciwpożarowymi;	ZAL	Dyskusja grupowa, prezentacje, projekty, zadania

					4. Zapoznanie się z polityką prywatności bezpieczeństwa, regułami ochrony danych osobowych i informacji (informacji niejawnych) oraz regułami obiegu dokumentów; 5. Analiza podstaw prawnych działania jednostki i jej właściwości; 6. Wykonywanie w warunkach rzeczywistych wybranych prac, zadań lub aktywności typowych dla kierunku kształcenia, wyszczególnionych w „Dzienniczku praktyki zawodowej”. II rok Warsztaty z praktyk realizowane w siedzibie Uczelni: 1. Pogłębianie wiedzy i umiejętności w ramach pracy zespołowej poprzez realizację wybranego projektu zespołowego związanego z kierunkiem studiów. Praktyka zawodowa realizowana u pracodawcy: 1. Wykonywanie w warunkach rzeczywistych wybranych prac, zadań lub aktywności typowych dla kierunku kształcenia, wyszczególnionych w „Dzienniczku praktyki zawodowej”; 2. Samodzielna realizacja przez studenta projektu dotyczącego firmy w której odbywa on praktykę zawodową pod nadzorem pracodawcy; 3. Komunikowanie się w pracy zawodowej poprzez właściwy dobór oraz stosowanie odpowiednich metod i narzędzi, w tym technik informacyjno-komunikacyjnych (ICT) oraz specjalistycznej terminologii; 4. Utrzymanie właściwych relacji w środowisku zawodowym oraz przestrzeganie zasad etyki zawodowej. III rok Warsztaty z praktyk realizowane w siedzibie Uczelni: 1. Wykonanie zadania praktycznego w postaci indywidualnego projektu dot. rozwiązania problemu w działalności i funkcjonowaniu danej organizacji związanego ze specjalnością wybraną przez studenta. Praktyka zawodowa realizowana u pracodawcy: 1. Wykonywanie w warunkach rzeczywistych wybranych prac, zadań lub aktywności typowych dla kierunku kształcenia, wyszczególnionych w „Dzienniczku praktyki zawodowej”; 2. Samodzielna realizacja przez studenta projektu z zakresu wybranej przez siebie specjalności pod nadzorem pracodawcy; 3. Wykorzystanie kontaktów ze specjalistami i pracownikami firmy do podniesienia umiejętności i kompetencji w zakresie wdrażania innowacyjnych rozwiązań związanych z wybraną specjalnością; 4. Utrzymanie właściwych relacji w środowisku zawodowym oraz przestrzeganie zasad etyki zawodowej;		

					5. Opracowanie raportu z odbytych praktyk zawodowych.		

Koordynator kierunku: ……

(podpis)

Dziekan Wydziału: Michalina Gryniewicz-Jaworska ……

(podpis)