POLITECHNIKA BYDGOSKA im. Jana i Jędrzeja Śniadeckich

WYDZIAŁ TELEKOMUNIKACJI, INFORMATYKI I ELEKTROTECHNIKI

PRACA DYPLOMOWA INŻYNIERSKA

na kierunku Informatyka Stosowana

Interaktywny konfigurator wnętrz pomieszczeń z dekoracyjnymi panelami 3D

Pracę wykonał: Antoni Malak

Nr albumu: 108650

Kierujący pracą: dr inż. Mariusz Sulima

Bydgoszcz, Styczeń 2023

Załącznik nr 1
do Regulaminu dyplomowania WTIiE

Politechnika Bydgoska
im. Jana i Jędrzeja Śniadeckich
Wydział Telekomunikacji, Informatyki i Elektrotechniki

KARTA PRACY DYPLOMOWEJ

Kierunek: Informatyka Stosowana

Forma studiów: stacjonarne ~~/ niestacjonarne~~

PRACA INŻYNIERSKA ~~/ MAGISTERSKA~~ **NR** 31/21/22/IST/SP

Student: Antoni Malak **Nr albumu:** 108650

Temat pracy (w języku polskim): Interaktywny konfigurator wnętrz pomieszczeń z dekoracyjnymi panelami 3D

Temat pracy (w języku angielskim): Interactive interior configurator of rooms with decorative 3D panels

Słowa kluczowe (w języku polskim): konfigurator wnętrz, wizualizacja 3D, panele dekoracyjne

Słowa kluczowe (w języku angielskim): interior configurator, 3D visualization, decorative panels

Zadania szczegółowe:

1. Przegląd istniejących rozwiązań w zakresie aranżacji wnętrz; 2. Wybór narzędzi i technologii umożliwiających przygotowanie realistycznej wizualizacji wnętrza pomieszczenia; 3. Określenie funkcjonalności umożliwiających interakcję takich jak wybór kształtu i wysokości pomieszczenia, umiejscowienia okien, wybór powierzchni do pokrycia panelami 3D, wybór paneli i kolorystyki; 4. Przygotowanie elementów 3D stanowiących wybór jednostek paneli 3D; 5. Zaprojektowanie logiki programowej konfiguratora umożliwiającej edycję przygotowanych elementów, zapisanie i powrót do projektu; 6. Realizacja konfiguratora wnętrz pomieszczeń z uwzględnieniem możliwości osadzenia go na stronie internetowej;

Miejsca przeprowadzenia prac:

- Zakład Systemów Teleinformatycznych

Kierujący pracą: dr inż. Mariusz Sulima dr inż. Mariusz Sulima
(Podpis)

Podpis studenta: Malak Antoni

Recenzent/Recenzenci (imię i nazwisko): dr inż. Łukasz Zabłudowski

PRODZIEKAN
ds. kształcenia i spraw studenckich
dr inż. Mścisław Srutek

Streszczenie

W ramach pracy zaprojektowany i zrealizowany został interaktywny konfigurator elementów wnętrz - ściennych paneli dekoracyjnych. Środowiskiem realizacji aplikacji jest platforma 3D Unity. Wirtualna przestrzeń została rozplanowana i zamodelowana od podstaw, włączając wszystkie elementy dekorowanych pomieszczeń. Na podstawie rysunków technicznych wymodelowane zostały meble i elementy pomieszczenia. Z powstałych elementów stworzone zostało pomieszczenie przedstawiające panele dekoracyjne z swoim naturalnym otoczeniu. Zaprojektowany i zaimplementowany został interfejs użytkownika integrujący rozproszoną logikę programu, z interaktywnymi elementami scenerii umożliwiający zmianę koloru prezentowanych paneli, ich tekstury, kształtu i ich ułożenia.

Słowa kluczowe: konfigurator wnętrz, wizualizacja 3D, panele dekoracyjne

Summary

As part of thesis, an interactive configurator of interior elements - wall decorative panels was designed and implemented. Application implementation was done using 3D platform Unity environment. Virtual space was planned and modelled from scratch, including all the elements of decorated room. All those elements were modeled using technical drawings as reference. From resulting elements, room was created it depicts decorative panels in their natural habitat. Graphical user interface was designed and implemented, it integrates distributed logic of the program with interactive elements of the scenery, allowing changing colors, textures, shape and laying of the panels.

Keywords: interior configurator, 3D visualization, decorative panels

Spis treści

1 Wstęp

Szybki rozwój oraz powszechność internetu pozwoliła na łatwiejsze dotarcie do potencjalnego klienta. Podając jedynie adres strony internetowej, zaprosić można kogoś na stronę internetową, której budowa i funkcje mogą być najróżniejsze. Może to być prosta statyczna strona wizytówka, której celem jest reklamowanie kogoś lub czegoś, lub skomplikowany profesjonalny program, który jest dostępny dla każdego.

Dawniejszą formą reklamy były bannery reklamowe i ulotki. Bannery zwracały uwagę masy osób, a ulotki pełniły rolę informacyjną, podsuwając klientowi bardziej szczegółowe informacje, dotyczące produktu. Niestety pewne rzeczy lepiej jest przedstawić w sposób graficzny, dlatego powstały katalogi, w których widać produkt w jego 'naturalnym środowisku'. Dzięki technologii możliwe jest dostarczenie podobnej reklamy poprzez internet, nie marnując przy tym papieru, oraz dając sobie jednocześnie więcej możliwości dzięki interaktywności.

Dzięki zaimplementowaniu funkcji interaktywnych w konfiguratorze, możliwe będzie rozszerzenie zestawu funkcji ze statycznego obrazu do namiastki wizyty w salonie sprzedającym ten produkt. Ważne jest odpowiednie podejście do wizualizacji. Możliwe będzie obejrzenie pojedynczego panelu, ale bardziej pomocny dla klienta będzie widok przykładowego wnętrza, z ich użyciem. Możliwa będzie częściowa edycja panelu, oraz pomieszczenia w którym będzie on przedstawiany.

1.1 Cel pracy

Celem pracy dyplomowej jest zaprojektowanie i realizacja interaktywnego konfiguratora paneli ściennych umożliwiającego wizualizację wnętrza pomieszczenia przestrzeni trójwymiarowej.

1.2 Wstępne założenia pracy

Możliwy będzie wybór panelu z katalogu, zmiana jego koloru, oraz zdefiniowanie sposobu ich ułożenia. Możliwa będzie również konfiguracja wnętrza, aby klient mógł obejrzeć produkt, przedstawiony we wnętrzu, które przypomina jego własny dom. Do wyboru będą z góry zdefiniowane pomieszczenia, pozwalające na częściowe dostosowywanie ich w kwestii kolorystyki wnętrza, ilości i rozmiaru okien oraz konfiguracji mebli. Ważnym aspektem będzie możliwość osadzenia programu na stronie internetowej, co zwiększy rzeszę potencjalnych klientów.

Przed napisaniem programu, wymagana będzie analiza podstawowych pojęć związanych z tematem, w celu lepszego zrozumienia tematu, oraz precyzyjniejszego nakreślenia zakresu funkcji, które powinien pełnić program. Trzeba będzie przeanalizować i porównać istniejące już rozwiązania, aby nowo powstały program nie powielał błędów swoich poprzedników i starał się inspirować ich mocnymi stronami. Warto byłoby rozłożyć ideę na czynniki pierwsze i zastanowić się nad różnymi ich aspektami, dzieląc te już istniejące na kategorie i porównując je ze sobą.

Dopiero potem możliwe będzie poprawne zdefiniowanie założeń i zakresu funkcji programu. Wyciągając wnioski z podobnych rozwiązań, będzie można zaprojektować rozwiązanie idealnie pasujące do postawionego problemu. Zdefiniowany zostanie zakres funkcjonalności, które powinna pełnić taka aplikacja. Ważne również będzie uprzednie zdefiniowanie stopnia interaktywności, gdyż implementacja pewnych funkcji nie może być możliwa w późniejszym stadium programowania. Warto również zastanowić się nad funkcjami jakie powinna pełnić internetowa aplikacja, która pełni również rolę reklamy.

Po fazach analizy i projektowania, możliwe będzie przejście do właściwej implementacji projektu, zaczynając od modelowania obiektów trójwymiarowych i wymaganych grafik, poprzez stworzenie interfejsu użytkownika, aż do zapro-

gramowania logiki programu, który będzie można osadzić na stronie internetowej.

2 Wstęp teoretyczny

Celem tego rozdziału jest wylistowanie i objaśnienie zagadnień których tyczyć będzie się ta praca inżynierska, ważne jest uprzednie zdefiniowanie pojęć za pomocą, których opisywać będziemy tworzony projekt.

2.1 Wizualizacja

Wizualizacja jest obrazową reprezentacją przedmiotu, procesu lub stanu. Dawniej odnosiło się to do obrazu, albo procesu jego tworzenia, w ludzkiej wyobraźni. Obraz ten istniał jedynie w naszej głowie, by przedstawić go komuś innemu wymagane było nakreślenie jego na jakimś medium, co wymagało dużych umiejętności i środków. Cała gałąź sztuki polega właśnie na doskonaleniu tego wymagającego procesu. [9]

Wizualizacja komputerowa zajmuje się czymś podobnym. Stara się przedstawić coś nie fizycznego za pomocą ilustracji. Przedstawiany może być przedmiot, proces albo inny bardziej abstrakcyjny koncept. Dzięki oprogramowaniu graficznemu możliwe jest zautomatyzowanie tego procesu dla, z góry określonych, scenariuszy. Zadaniem wizualizacji komputerowej jest zebranie ogromu danych, odpowiednie ich przetworzenie, oraz przedstawienie ich w prosty i zrozumiały sposób użytkownikowi końcowemu.

Medium komputerowe pozwala na wiele możliwości w kwestii edycji prezentowanej grafiki. Parametry mogą zostać łatwo zmienione, a ich zmiana może zostać odzwierciedlona w warstwie graficznej. Mogą się one tyczyć działania całego programu i jego algorytmów, albo części renderującej grafikę. Peryferia komputerowe pozwalają na łatwe wprowadzanie i dostrajanie parametrów, a program bierze te zmienione wartości pod uwagę tworząc nowy, zmieniony obraz. Funkcja pozwalająca na wprowadzanie zmian w symulacji, oraz natychmiastowa jej reakcja jest podstawą interaktywności.

2.2 Interaktywność

Interakcja jest wzajemnym oddziaływaniem na siebie osób, przedmiotów lub zjawisk.[4] Ważne żeby oddziaływanie te było dwu-kierunkowe, gdyż jednokierunkowym odpowiednikiem interakcji jest przyczynowość. Interaktywność jest miarą interakcji i jej jakości.

Interaktywność jest sytuacją w której każda kolejna akcja bierze poprzednią, lub kilka poprzednich, pod uwagę zmieniając jej kontekst. Wykonanie tej samej akcji w innym kontekście będzie skutkować wykonaniem innej czynności. Ważne jest aby różne konteksty były łatwo rozpoznawalne i znane użytkownikowi.

Interaktywność może zostać zdefiniowana jako ciągła wymiana informacji, pomiędzy minimum dwoma uczestnikami. W tym przypadku jednym uczestnikiem jest wizualizacja, a drugim użytkownik. Użytkownik podaje informacje w postaci parametrów wejściowych, a wizualizacja zwraca użytkownikowi jak wyglądać może scena spełniające podane wymagania i parametry. Jest to proces który zachodzi cały czas w formie pętli, lecz nie każda akcja użytkownika musi równać się natychmiastowej reakcji systemu. [8]

Ważne jest zachowanie przejrzystości po obu stronach. Należy dać użytkownikowi interfejs, który jest zrozumiały i łatwy w obsłudze. Dzięki zastosowaniu się do podstawowych zasad projektowania interfejsów powinien on być wizualnie przejrzysty i jednolity. Sensowne rozplanowanie różnych kontekstów, pozwoli na bardziej intuicyjną nawigację po programie i jego funkcjach. [12]

Interaktywność pozwala użytkownikowi na kontrolowanie parametrów wejściowych symulacji, oraz obserwowanie nowo powstałego obrazu. [7] Ważne jest aby użytkownik czuł, że ma kontrolę nad przebiegiem symulacji, a każda

wprowadzona zmiana jest wprowadzana i aktualizowana tak szybko jak szybko się da. Dlatego czas obliczenia kolejnego widoku powinien być odpowiednio krótki. Zawęża to częściowo jak wiele obliczeń możemy przeznaczyć na ten cel, oraz jak wymagająca i realistyczna może być wizualizacja i jej grafika.

2.3 Grafika komputerowa

Grafika komputerowa jest dziedziną zajmującą się tworzeniem, przetwarzaniem oraz wyświetlaniem obrazów w medium komputerowym. Obraz może być wygenerowany przez komputer, tworząc obraz generowany komputerowo (tzw. CGI), albo może być odzwierciedleniem prawdziwego fizycznego obiektu. Do stworzenia zdjęcia obiektu używane są matryce światłoczułe, zmieniające docierające do niej fotony, na sygnał cyfrowy. Możliwe jest również stworzenie obrazu, będącego wizualizacją powstałą z pomiarów innych urządzeń i sensorów cyfrowych. Mogą to być zarówno wykresy przedstawiające zmianę wartości mierzonej przez sensor, albo coś bliższego obrazowi obiektu jak trójwymiarowy model powstały z zeskanowania bryły fizycznego obiektu.

Podstawowe biblioteki graficzne, takie jak SDL, OpenGL, pozwalają na rysowanie podstawowych kształtów, obrazów i stosowanie podstawowych przekształceń na tych obrazach. Może to być wystarczające do stworzenia prostych wizualizacji, ale aby osiągnąć oczekiwane wyniki wymagany byłby ogromny wkład pracy. Biblioteki te nie pozwalają na wyświetlanie skomplikowanych modeli trójwymiarowych, ani ich tekstur. Oświetlenie musiałoby być zaimplementowane od podstaw wymagając ogromnej wiedzy oraz dobrej optymalizacji. Na szczęście istnieje wiele istniejących już rozwiązań, które implementują wszystkie wymagane funkcje, są to silniki gier komputerowych.

2.3.1 Silniki gier komputerowych

Silnik gry komputerowej nie obejmuje jedynie warstwy wizualnej, jest to kompletny pakiet obejmujący każdy aspekt interaktywnego programu z naciskiem na generowanie grafiki w czasie rzeczywistym. Ważniejszymi częściami silnika, użytymi w projekcie będzie pobieranie inputu od użytkownika, modyfikacja modeli i sceny w czasie rzeczywistym, oraz wyświetlenie nowo wygenerowanego obrazu.

Sterowanie odbywa się poprzez urządzenia peryferyjne, takie jak klawiatura, myszka czasami ekran dotykowy, albo tablet graficzny. Możliwe jest również użycie niektórych wewnętrznych sensorów urządzenia takich jak żyroskop czy akcelerometr, które pozwalają na detekcję orientacji urządzenia mobilnego. Pozwalają one na dokładne i precyzyjne manipulowanie sceną. Jest to Kolejny mechanizm, który został już zaimplementowany jako część silnika, oraz nie wymaga on implementacji.

Jedną z funkcji silnika jest możliwość łatwego tworzenia interfejsu użytkownika, za pomocą którego użytkownik może sterować zachowaniem programu. Interfejsy mogą składać się z warstw, mogą pełnić rolę informacyjną (wskazywanie stanu zmiennych), albo interaktywną (wirtualne przyciski). Niektóre akcje programu mogą być wykonywane w sposób mniej jawny, ale bardziej intuicyjny (n.p. gesty dotykowe do obracania widoku). Inne natomiast będą wymagały kliknięcia na przycisk, albo inny element interfejsu.

Dzięki gotowemu silnikowi graficznemu, zamiast implementować silnik graficzny od zera, możliwe będzie zaimportowanie modeli wraz z teksturami. Modele mogą być oteksturowane, oraz mogą być zrobione z różnego rodzaju materiałów, które inaczej wyglądają i wchodzą w interakcję z oświetleniem. Do sceny można dodawać różne rodzaje obiektów, mogą to być proste modele trójwymiarowe, ale również różne rodzaje źródeł światła, albo efekty cząsteczkowe. Wymagane będzie również ręczne ustawienie sceny, oraz wszystkich jej elementów. Sceny mogą posiadać elementy in-

teraktywne, co musi być uwzględnione w kompozycji jak i implementacji.

2.3.2 Znane silniki gier komputerowych

Rynek oferuje wiele gotowych rozwiązań, w kwestii silników do gier komputerowych. Mimo że one wszystkie mają ten sam cel, skupiają się one na różnych aspektach oraz posiadają pewne unikalne cechy. Jedne skupiają się na realistycznej grafice oraz stremowaniu ogromnych ilości danych, a inne skupiają się na prostocie użycia oraz multiplatformowości. Każdy z nich swoimi możliwościami określa pewien zakres, w którym gra będzie się zawierać. [6] Pełna gra stworzona na danym silniku nie może wyglądać lepiej niż demo technologiczne czy benchmark tego samego silnika, gdyż po prostu zabraknie zasobów na resztę logiki gry.

Unreal engine

Silnik graficzny stworzony dla gry Unreal przez studio Epic Games. Gra Unreal została wydana w 1998 roku i była jedną z pierwszych trójwymarowych gier FPS. Była to w pełni gra trójwymairowa, która nie używała dwuwymiarowych sprite-ów zamiast przeciwników, jak wcześniej, a modeli trójwymiarowych. Były to czasy początków akceleratorów graficznych, dlatego niektóre obliczenia były wykonywane na procesorze, a inne za pomocą akceleratora graficznego.

Zestaw narzędzi był o wiele bardziej podstawowy niż te spotykane obecnie ze względu na ograniczenia graficzne oraz fakt, że gry komputerowe nadal były nowością, a technologia do tworzenia gier komputerowych była bardzo wczesna. Dlatego zostały w nim zaimplementowane funkcje, które można uznać za dosyć podstawowe, takie jak: detekcja kolizji, filtrowanie tekstur, wsparcie dla kolorowych źródeł światła, oraz edytor poziomów. Ciekawą nową funkcją było dodanie obiektów wchodzących w interakcję ze światłem, źródeł światła, przestrzennej mgły, czy implementacja sklepienia nieba.

Z czasem silnik się rozwijał, oddawane zostały kolejne narzędzia, a większa moc obliczeniowa komputerów pozwalała na stworzenie realistycznej grafiki. Czwarta wersja dodawała nowy sposób na generowanie oświetlenia, zamiast statycznego oświetlenia sceny użyte było dynamiczne generowanie oświetlenia, reagujące na poruszające się obiekty jak i samego bohatera. W tej samej wersji dodano również narzędzia do szybkiej implementacji logiki gry.

Najnowsza piąte wersja, dodała przełomową funkcję, silnik Nanite. Pozwala on na importowanie modeli w najlepszej możliwej jakości oraz zmniejszanie ich jakości w locie, wedle potrzeby. Jeśli obiekt zajmuje większą powierzchnię ekranu, model rysowany jest z większą ilością wielokątów, jeżeli jest mniej widoczny jest on używany z mocno ograniczoną ich liczbą. [5] Z perspektywy programisty jest to wygodne i nie wymaga ręcznej implementacji LOD, oraz jednocześnie samo dba o optymalizację modeli w całej grze.

Wymagania sprzętowe silnika są wysokie, mogą one być zbyt wysokie dla niektórych komputerów. Celem jest stworzenie wizualizacji, którą można będzie uruchomić na większości urządzeń, ale nie można wymagać od użytkownika posiadania nawet dedykowanej karty graficznej. Silnik byłby idealny jeżeli głównym celem byłaby realistyczna grafika, ale niestety jest on narzędziem o zbyt wysokich wymaganiach sprzętowych, których niektóre urządzenia klientów nie mogłyby sprostać.

Unity3D

Unity jest silnikiem gier stworzonym na potrzeby tworzenia gier dla systemu Mac OS. Potem został zmodyfikowany do wpierania innych platform takich jak: komputery stacjonarne, konsole oraz platformy mobilne. Dodane zostało

również wsparcie dla gier wirtualnej rzeczywistości. Z jego pomocą możliwe jest stworzenie gier zarówno dwuwymiarowych jak i trójwymiarowych. Znany jest ze swojej prostoty i łatwości obsługi.

Silnik nie jest przeznaczony do tworzenia gier AAA, lecz jest nastawiony na szybkie i łatwe tworzenie prostszych gier nastawionych na ciekawą rozgrywkę, albo jej mechanikę. W tego typu grach posta grafika nie jest problemem, dlatego często jest on używany do tworzenia niezależnych gier komputerowych. Często również jest on używany do tworzenia gier mobilnych, gdzie szybkość tworzenia gier jest dosyć ważna, by móc nadążać za nieustannie zmieniającymi się trendami.

W obecnej wersji silnika, gry programowane mogą być w języku C#. Pozwala to na lepsze użycie obiektowości oraz unikalnych funkcji języka C# nastawionych na programowanie obiektowe. Środowisko programistyczne do tworzenia gier jest dostępne, dla systemów operacyjnych Windows, GNU/Linux oraz Mac OS, a tworzone projekty mogą być przeznaczone na każdą możliwą wspieraną platformę.

Możliwe jest stworzenie gry wieloosobowej. Dzięki zastosowaniu wielu wirtualnych kamer możliwe jest zaimplementowanie gry wieloosobowej na podzielonym ekranie. Silnik posiada warstwę sieciową, dzięki której można tworzyć sieciowe gry wieloosobowe, pozwalające na grę dwóch lub więcej graczy przez internet. Pozwalają one również na tworzenie serwerów gry wieloosobowej.

Liczba edytorów i narzędzi jest mniejsza i mniej wyspecjalizowana dlatego ten sam zestaw narzędzi można wykorzystać w większości projektów, niezależnie od gatunku tworzonej gry. Można tworzyć animacje dwuwymiarowe oraz trójwymiarowe, a wbudowany edytor animacji, pozwala na płynne przełączanie się pomiędzy różnymi animacjami.

Wbudowany edytor interfejsu użytkownika pozwala na tworzenie interaktywnych ekranów, z różnego rodzaju przyciskami, suwakami oraz innymi wirtualnymi metodami wprowadzania i wyświetlania danych. Pozwalają one na stworzenie zarówno interfejsu użytkownika widocznego podczas gry, jak i bardziej złożonych ekranów wyboru, czy ekranów sterowania ustawieniami gry.

Silnik fizyczny posiada podstawowe funkcjonalności, pozwalające na tworzenie gier opartych o fizykę. Obiekty gry mogą posiadać wagę, wyporność, oraz implementować cechy fizycznych materiałów z których są zrobione. Fizyczne obiekty mogą być połączone ze sobą wiązaniami, imitującymi połączenia przedmiotów w prawdziwym świecie, albo te zupełnie wirtualne. Możliwe jest emulowanie działania zawiasu, lin, szyny czy sprężyn. Wszystkie z nich są sparametryzowane co pozwala na ich dostrojenie, a nawet tworzenie własnych połączeń fizycznych.

Zintegrowanym środowiskiem programistycznym na platformie Windows i Mac OS, jest Visual Studio, nie jest ono dostępne na systemy operacyjne GNU/Linux. Istnieje wsparcie w postaci dodatków i wtyczek dla Visual Studio Code, które pozwalają na wygodne pisanie kodu i skryptowania na wszystkich wymienionych platformach.

Godot

Godot jest silnikiem na licencji otwartego oprogramowania, co czyni go w pełni darmowym. Został stworzony przez argentyńskich programistów na potrzeby gier komputerowych w Ameryce Łacińskiej. Wspiera on większość znanych systemów operacyjnych: Windows, macOS, Linux oraz różne wersje BSD. Dodatkowo wspiera on platformy mobilne, oraz uruchamianie gier w przeglądarce. Jest przeznaczony do tworzenia małych i średnich gier, oraz grafika przez niego generowana nie jest w stanie dorównać tej, reprezentowanej przez obecne gry AAA.

Gry można programować w wielu językach, wspierany jest unikalny język skryptowy GDScript, napisany na potrzeby silnika, oraz język c++ i c#. Dodatkowo dzięki wsparciu dla wiązań językowych możliwe jest pisanie kodu w językach ta-

kich jak: Rust, Nim, JavaScript, Haskell, Clojure, Swift. Dodatkowo możliwe jest programowanie wizualne, pozwalające na składanie logiki programu, bez znajomości języków programowania. Zintegrowany edytor tekstu, wspiera podstawowe funkcje, środowiska programistycznego takie jak: formatowanie kodu, podświetlanie pisowni, uzupełnianie kodu, wbudowany jest również debugger pozwalający na tworzenie punktów wstrzymania i uruchamianie programu, linijka po linijce.

Wspierane jest tworzenie gier dwuwymiarowych i trójwymiarowych, gdzie oba tryby mogą pracować jednocześnie, oddzielnie od siebie nawzajem. Stworzony został system animacji, dla obrazów dwuwymiarowych i modeli trójwymiarowych. Animacja szkieletowa jest wspierana w obu przypadkach, animacje obrazu posiadają dodatkowe unikalne dla obrazów animacje. Obrazy można skalować, obracać i edytować ich kolorystkę w czasie trwania animacji.

Kolejnymi wspieranymi elementami jest tworzenie graficznego interfejsu użytkownika. Czasami ten sam silnik używany jest właśnie, do tworzenia programów użytkowych, zamiast gier komputerowych. Jednym ze znanych programów z jego użyciem jest program Material Maker.[2] Wspierana jest również, wielowątkowość, statyczne oświetlenie sceny, oraz efekty cząsteczkowe.

Godot przeznaczony jest to tworzenia małych i średnich projektów, oraz daje użytkownikowi wiele możliwości. Jest rozwiązaniem na licencji otwartego oprogramowania, dzięki czemu posiada duże wsparcie wolontariuszy. Jest to dobry wybór do stworzenia projektu wizualizacji architektonicznej.

CryEngine

Jest silnikiem zaprojektowanym na potrzeby gry Far Cry, przez firmę Crytek. Jego kolejna iteracja została użyta to gry Crysis, która jest znana z bardzo realistycznej grafiki, oraz jest uznawana za jeden z największych przeskoków graficznych i technologicznych w historii gier trójwymiarowych. Silnik ten wyprzedzał swoje czasy oraz pozwalał na generowanie realistycznej grafiki i posiadał fizykę, która pozwalała na niespotykaną dotąd interakcję z otoczeniem.

Silnik pozwala na tworzenie gier AAA, z wymagającą i realistyczną grafiką. Był tworzony z nastawieniem na gry FPS, co odbija się w doborze narzędzi do niego dołączonych. Tworzenie postaci jest proste dzięki narzędziom do ich animacji. Istnieje osobny system do szkieletowych animacji postaci, a animacje te mogą być parametryzowane, co pozwala na dogłębniejsze dostrajanie ich. Przydatnym dodatkiem jest też osobny edytor animacji twarzy.

W tworzeniu rozgrywki FPS, pomaga generacja kuloodpornych zasłon, oraz częściowo zniszczalne otoczenie. Niektóre obiekty mogą być zniszczalne, a nawet po zniszczeniu nadal być poddawane fizyce gry. Elementami fizycznymi w grze mogą być również wszelkiego rodzaju liny, pozwalające na łączenie obiektów, oraz tworzenie zniszczalnych mostów. Dynamiki rozgrywce dodać może dynamiczne wyszukanie ścieżek, pozwalające przeciwnikom na zbliżenie się do postaci gracza.

Wsparcie również dostały pojazdy, których poruszanie się można łatwo zaimplementować, oraz też posiadają dedykowane sobie narzędzia. Edytor pojazdów pozwala na edycję oraz dostrojenie parametrów istniejących pojazdów, albo stworzenie nowego. Systemy pomagające w tworzeniu pojazdów, uzupełnia system budowania dróg na mapie, dzięki któremu w szybki sposób można narysować i wydzielać drogi na mapie świata gry.

Przez lata zostało stworzonych wiele narzędzi, które ułatwiają tworzenie gier w wielu aspektach. Pozwalają na monitorowanie wydajności i parametrów gry podczas pracy, tworzenie materiałów i tekstur, obszerna baza gotowych modeli trójwymiarowych. Istnieje osobna grupa narzędzi pozwalająca na tworzenie i animowanie postaci. Istnieją, też narzędzia przeznaczone do efektów dźwiękowych oraz ich odpowiedniego rozmieszczenia w przestrzeni gry.

2.4 Interaktywny konfigurator produktu

Konfigurator, dawniej nazywany konfiguratorem produktu, jest narzędziem do tworzenia unikalnej konfiguracji produktu posiadającego wiele opcji konifguracyjnych. Produkty te posiadają wiele opcji wyboru, niektóre opcjonalne, inne wymagane, a inne wykluczające się nawzajem. Pozwala on na dogłębne dobranie i skonfigurowanie produktu, odpowiadającego klientowi, gdzie efektem końcowym jest gotowa konfiguracja, kompatybilna z systemem. Może on być przeznaczony dla samego klienta, albo być narzędziem wprowadzania dla pracownika jego obsługującego. Ma on na celu oddelegowanie zadania tworzenia konfiguracji z pracownika, na program komputerowy i samego klienta. [3]

Dla klienta może on być interaktywnym katalogiem pokazującym zbiór produktów, oraz wszelkich ich opcji konfiguracyjnych. Czasami klient do końca nie sprecyzował jeszcze swojej wizji i wymagań, ani jak produkt końcowy miałby wyglądać. Poprzez kolejne kroki tworzenia kompozycji, klient jest zapoznawany z zakresem możliwości edycji i opcji dostrojenia produktu końcowego. W ten, koherentny, sposób udziela klientowi informacji, niezbędnych do podjęcia najlepszego dla siebie wyboru. Na końcu informacja ta może być w zautomatyzowany sposób przekazana do realizacji.

Dla pracownika, może być narzędziem pomagającym w pracy i pozwalającym na wprowadzenie gotowej, zwalidowanej konfiguracji do systemu. Może przeprowadzić on pracownika przez kolejne etapy, oraz poinformować go o stanie magazynu, albo innych niedogodnościach, dotyczących łańcucha dostaw. Dzięki temu, pracownik jest odciążony, a swoją uwagę może skupić na zadawaniu bardziej szczegółowych pytań klientowi.

2.4.1 Typy konfiguratorów

Ze względu na zastosowaną technologię i grafikę, wydzielić można typy konfiguratorów produktów. Nie jest to jedyny sposób w jaki można je podzielić.

Prerenderowane wizualizacje

Są to wizualizacje złożone z wygenerowanych wcześniej obrazów. Zdjęcia oraz prerenderowane obrazy charakteryzują się najlepszą jakością grafiki, oraz są zasobo-oszczędne (z punktu widzenia użytkownika). Zdjęcia jednak nie mogą być mocno zmodyfikowane po ich zrobieniu, zazwyczaj zmieniane są kolory elementów, albo niektóre elementy są zmieniane na inne. Każda wizualna konifguracja oraz jej możliwości muszą zostać określone z góry, oraz wzięte pod uwagę.

Technologie interaktywne

Technologie cechującą się większą interaktywnością, pozwalające na obracanie obrazu lub przedmiotu, oraz większe edycje otoczenia. Są one niestety najbardziej wymagające pod względem zasobów, oraz nie zawsze urządzenie klienta może tym wymaganiom sprostać. Do ich stworzenia używane są czasami silniki do gier komputerowych.

2.5 Architektura wnętrz

Architektura wnętrz jest dziedziną zajmującą się projektowaniem oraz wystrojem wnętrz budynków. Obejmuje to zdefiniowanie stylu, i ogólnej kolorystyki wnętrza jak i rozplanowanie rozmieszczenia mebli. Odpowiednie, do jego zastosowania, zagospodarowanie wnętrza sprawia, że wnętrze jest bardziej funkcjonalne, oraz prezentuje się lepiej. [10] W kontekście tej pracy wymagane będzie stworzenie zestawu projektów pomieszczeń, w taki sposób, aby najlepiej oddawały

najróżniejsze kompozycje i style pomieszczeń. Zbiór ten powinien oddawać jak najszerszą gamę stylów, aby klient mógł w nim znaleźć coś co mniej więcej oddaje jego własne wnętrze, aby lepiej do niego dopasować swój własny produkt.

2.5.1 Istniejące już rozwiązania

Trudno znaleźć rozwiązanie, które byłoby używane przez więcej niż jedną firmę. Zazwyczaj każda firma z kolei tworzy własną implementację od podstaw. Przyczyną może być różnorodność produktów, oraz jego otoczenia. Każda osobna implementacja musiałaby posiadać swoje własne tekstury, pomieszczenia, modele. Czasami nawet zakres edycji i rozmieszczenia kamery byłyby zupełnie inne. Ilość włożonej pracy jest niewiele mniejszy niż zaprogramowanie własnego rozwiązania od podstaw. Właśnie dlatego każda branża z kolei decyduje się na tworzenie własnej, skrojonej na miarę, implementacji. Wydzielić można pewne części wspólne i funkcjonalności, które mają niektóre z nich.

2.5.2 Interaktywny widok trójwymiarowy

Widok trójwymiarowy pozwalający na ograniczone obracanie kamery, oraz oglądanie produktu w swoim naturalnym otoczeniu. Pozwala to klientowi na bardziej wizualne podejście do decyzji, oraz skupienie się na innych aspektach i wizualnych cechach produktu, takich jak kolor, rozmiar czy jego kształt. Pokazanie produktu blisko przedmiotów o wiadomych rozmiarach, pozwala na lepszą ocenę rozmiaru produktu, który jeszcze fizycznie nie istnieje.

2.5.3 Widok projektowania

Widok przedstawiający produkt w uproszczonej formie, przypominający rysunek techniczny. Pozwala na dostosowanie wymiarów produktu do wymiarów istniejącego już pokoju lub zabudowy. Używany jest kiedy wymiary otoczenia są znane, albo wymiary produktu mogą się mocno różnić zależnie od dobieranego typu lub modelu. Jego zaletą jest przejrzystość, oraz uproszczony, czysty wygląd pozwalający się skupić na jego wymiarach.

2.5.4 Katalog produktów i jego opcji

Częścią konfiguratora może być osobny widok, albo okno pokazujące użytkownikowi inne produkty, albo inne jego wersje. W ten sposób pokazujemy klientowi asortyment i jakie on ma inne możliwości do wyboru. Tego typu widok może pokazywać różne warianty rozmiarowe, stylistyczne czy kolorystyczne produktu. Kolejne wybory można reprezentować poprzez miniaturowy obraz, model trójwymiarowy, albo opis tekstowy, a nawet krótki tytuł. Ważne jest aby kolejne wybory były ustawione w odpowiedniej kolejności, od najogólniejszego do najbardziej szczegółowego.

2.5.5 Ekran podsumowania

Ekran wypisujący wszystkie dokonane zmiany, oraz opcje dodatkowe. Ma na celu pokazanie klientowi ceny produktu końcowego oraz przekazanie innych ważnych dla niego informacji, takich jak czas realizacji, dane kontaktowe oraz numer zamówienia. Po ostatnim kliknięciu, konifguracja produktu powinna zostać dodana do systemu oraz przetworzona.

2.6 Panele ścienne

Panele są płaskimi fragmentami materiału, o jednakowych kształtach, którymi wykłada się podłogi, sufity oraz ściany budynków. Na początku robiło się to w celu odizolowania termicznego, kamiennych ścian i podłóg. Wraz z czasem,

panele zaczęły pełnić rolę dekoracyjną. Wykonywane one były na początku z drewna, wraz z rozwojem technologii i wynalezieniem nowych materiałów zaczęto je tworzyć z tworzyw sztucznych. Dodatkowo wyłożenie powierzchni miększym materiałem, zmniejsza echo w pomieszczeniu. Efekt ten jest użyty w celu całkowitego wytłumienia echa, za pomocą specjalnych paneli wygłuszających. Paneli można również używać izolacji termicznej budynku, tzw. ocieplania budynku.

2.6.1 Tapety

Tapety są cienką okładziną naklejaną na wnętrza budynków w celach dekoracyjnych, wykonane głównie z papieru, ale również tkanin, skóry i tworzyw sztucznych. Dzięki swojej elastyczności i cienkości mogą być zwijane w rolki na czas transportu, co znacznie ułatwia ich transport. Kolejnym czynnikiem jest ich bardzo mała waga w przeciwieństwie do płytek i paneli nie wymagają mocnego kleju do instalacji. Przykleja się je do ściany za pomocą specjalnego kleju do tapet.

Tapety mogą mieć różne kolory i wzory, dzięki ich różnorodności mogą w duży stopniu zmienić sposób w jaki postrzegane jest pomieszczenie. Jaśniejsze kolory wizualnie powiększają pomieszczenie, a ciemniejsze je zmniejszają. Pionowe wzory sprawiają, że pomieszczenie wydaje się być wyższe, a poziome poszerzają je. Dodatkowo mogą one posiadać wytłoczenia, oraz własną teksturę, w ten sposób podkreślając swój wzór, albo używając jednolitego koloru, oraz swojego własnego wytłoczenia jako subtelnego wzoru.

2.6.2 Panele wygłuszające

Wytłumianie pomieszczenia polega na zmniejszaniu echa w pomieszczeniu poprzez pokrywanie jego ścian izolatorami akustycznymi. Jest to materiał, który dzięki swoim właściwościom, zapobiega niepożądanemu przenikaniu dźwięków oraz odbijaniu ich tworząc echo. Zależnie od zastosowania materiały te mogą być umieszczone wewnątrz ścian budynku, albo na wewnętrznej stronie ścian pomieszczenia. Umieszczenie ich wewnątrz ograniczy przedostawanie się dźwięków z wnętrza pomieszczenia do zewnątrz i na odwrót, a umieszczenie ich na wewnętrznej stronie ścian, efektywniej zmniejszy echo wewnątrz pomieszczenia.

Izolatory akustyczne mogą występować w różnych formach. Jeżeli jego grubość na to pozwala, może to być giętki materiał sprzedawany w rolkach, który można przyklejać do ścian jak tapetę. Wariant o większej grubości jest sprzedawany w formie jednakowych paneli na ścianę, które posiadają falowane wyżłobienia polepszające ich właściwości tłumiące. Są one czasami sprzedawane w różnych wariantach kolorystycznych oraz pozwalają na tworzenie kolorowych wzorów na ścianę. Ostatnim typem są struktury tłumiące takie jak pułapki dźwiękowe, są to zazwyczaj pudła o różnych wymiarach, zazwyczaj wykonane z drewna z odpowiednio wywierconymi dziurami i szczelinami. Pełnią one rolę odwrotną do pudła rezonansowego, łapiąc i wytłumiając całkowicie dźwięki.

2.6.3 Panele dekoracyjne

Inną funkcją jaką mogą pełnić panele ścienne jest dekoracja wnętrza, ich zadaniem jest przełamanie monotonii dużych jednolitych ścian w pomieszczeniu. Dodatkowo, są one mniej inwazyjne dla ściany niż tapeta lub glazura, której tak łatwo nie można wymienić na nową. Rozwój technologii i materiałoznawstwa pozwala obecnie na tworzenie klejów i mocowań wystarczająco mocnych na przytwierdzanie paneli do ściany, bez wiercenia dziur w ścianie, ani zostawiania śladów kleju na niej.

Ważne aby panele miały specjalny kształt, pozwalający na szczelne pokrycie powierzchni nazwaną parkietażem. Parkietaż formeny składa się z identycznych wielokątów foremnych, w którym jednakowa liczba figur, schodzi się w wierzchołku. W praktyce oznacza to, że wyprodukować będzie jedynie panele o w kształcie jednej figury geometrycznej. Trzema znanymi wielokątami foremnymi tworzącymi parkietaż foremny są kwadrat, trójkąt równoboczny oraz sześciokąt foremny. Nie trzeba się ograniczać parkietażu foremnego, dlatego możliwe jest użycie innych kształtów, które nie są foremnymi wielokątami.

2.7 Atuty prezentacji treści w formie strony internetowej

Dzięki osadzeniu aplikacji na stronie internetowej, będzie ona dostępna dla ludzi z całego świata, co pozwoli na dojście do większej liczby klientów. Każdy kto zna adres domeny, będzie mógł ze swojego urządzenia, smartfona czy komputera, odwiedzić stronę i skorzystać z konfiguratora. Po przejściu przez wszystkie etapy, podsumuje ona zamówienie, oraz wyśle je na serwer wraz ze wszystkimi niezbędnymi informacjami. Automatyzuje to proces zapoznania klienta z ofertą, podjęcia decyzji, oraz złożenia zamówienia.

Konfigurator jest jedynie częścią strony internetowej, oraz pełni inny zakres funkcji niż ona sama. Reklamowa strona internetowa, zazwyczaj pełni rolę internetowej wizytówki, udzielając klientowi informacji na temat zakresu obowiązków firmy, katalogu produktów, oraz lokacji i innych informacji kontaktowych. Z czasem zaczęto rozwijać ten pomysł pozwalając użytkownikowi na wchodzenie w interakcję z firmą, za jej pomocą. Pozwalać może na przykład na wysłanie wiadomości do pracowników firmy, albo skorzystanie z czatu internetowego.

2.7.1 Konfigurator jako katalog produktów

Konfigurator może w inteligentny sposób przejąć niektóre funkcje informacyjne strony, albo przedstawić je w bardziej przystępny sposób. Dotychczas katalog produktów przedstawiany był w formie listy lub broszury ze zdjęciami produktu, jako coś statycznego. Dzięki interaktywnemu podejściu do problemu, można w bardziej przystępny sposób przedstawić zakres świadczonych usług, przedstawić poszczególne produkty, oraz pozwolić na dostosowanie go do własnych potrzeb.

3 Analiza i założenia projektu

Przez implementacją programu należy rozplanować jego wszystkie aspekty, zaczynając od spraw najbardziej ogólnych do rzeczy bardziej szczegółowych. Należy zdefiniować wymagania, które ma on spełniać, oraz zakres funkcji które będzie on implementował. Im bardziej szczegółowo zostanie to rozpisane, tym mniej zostawione zostanie miejsca na niedopowiedzenia i potencjalne błędy logiczne.

3.1 Wymagania sprzętowe

Ważnym aspektem programu jest dotarcie do jak największej grupy odbiorców, dlatego program powinien być dostępny oraz osiągalny dla jak największej liczby potencjalnych klientów. Jest wiele czynników, które mogą ograniczać dostęp, a jednym z nich są wymagania sprzętowe urządzenia klienta. Zakres urządzeń których mógłby używać klient jest bardzo obszerny, ale można założyć że program powinien płynnie pracować na komputerach biurowych ze zintegrowaną kratą graficzną oraz smartfonami z niskiej półki cenowej.

Program będzie uruchamiany z przeglądarce internetowej, dlatego powinien on wspierać najbardziej popularne z nich, takie jak Google Chrome, Safari, Edge oraz Mozilla firefox. Warto nadmienić, że niektóre z tych przeglądarek internetowych, posiadają swoje odpowiedniki na platformy mobilne, dla których również powinna zostać przygotowana mniej wymagająca wersja programu. Wybrany do realizacji projektu silnik Unity 3D wspiera tworzenie osobnych zestawów ustawień, przeznaczonych na urządzenia o różnych osiągach i mocy obliczeniowej. Niezbędną funkcją każdej z tych przeglądarek jest wsparcie dla renderowania OpenGL, gdyż jest ono używane przez silnik używany przez Unity3D.

Program może wymagać dosyć skomplikowanego sterowania, wymagany będzie wybór opcji, które mogą być reprezentowane za pomocą ikon oraz tekstu, oraz dostrajanie wymiarów produktu. Sterowanie będzie odbywać się poprzez ekran dotykowy smartfona lub tabletu, albo poprzez klawiaturę i myszkę komputera stacjonarnego. Wymagane jest stworzenie schematu sterowania który może być wygodnie i intuicyjnie używany z użyciem obu metod wprowadzania.

Zazwyczaj programy tworzy się z myślą o konkretnych docelowych systemach operacyjnych. W sprzyjających warunkach, można dostosować program do pracy innym systemie operacyjny, ale zależy do od wielu czynników i nie zawsze może być możliwe. Silnik Unity3D jest wspierany na większości systemów operacyjnych, dlatego projekt stworzony z jego użyciem będzie również wspierany przez te systemy.

3.2 Zakres funkcjonalności

Użytkownik zostanie przeprowadzony przez kolejne etapy konfigurowania produktu. Kolejne wybory będą stopniowo zawężać oraz prowadzić go do podjęcia najlepszej decyzji. Ważna jest ich kolejność, pierwsze -wybory mają mieć większe znaczenie, a każdy kolejny w coraz mniejszym stopniu będzie w stanie zmieniać ogólny zarys produktu.

Wybrać będzie można kolorystkę wnętrza, do wyboru będzie parę naście palet, które zmienią kolory ścian, mebli oraz niektóre tekstury pomieszczenia. Dzięki temu, użytkownik, będzie można w pewnym stopniu odwzorować kolorystykę swojego własnego wnętrza, do którego chce dobrać panele dekoracyjne.

Pierwszym wyborem dotyczącym paneli będzie wybór ich kształtu. W osobnym widoku będzie można przeglądać gotowe kształty i style paneli dekoracyjnych. Dostępne będą trójkątne, kwadratowe, sześciokątne, oraz inne bardziej skomplikowane formy. Wszystkie panele będą miały ten sam kształt.

Domyślnie panele będą poustawiane jeden przy drugim, ale możliwe będzie ich jednolite rozstąpienie, tworząc jed-

nolite marginesy pomiędzy nimi. Wprowadzić będzie można dokładną szerokość odstępów między nimi. Możliwym wyborem będzie również obrócenie wszystkich paneli o 30 albo 45 stopni aby zmienić ich ułożenie na alternatywne.

Wybrać również będzie można kolor paneli. Do wyboru będzie gotowy zbiór kolorów do wyboru. Panele będą miały jednolity kolor. Zależnie od ilości miejsca na danej powierzchni i jej wymiarów, wybrać będzie można ilość rzędów oraz kolumn. Pozwoli to pokryć powierzchni o różnych wymiarach, panelami tej samej wielkości.

Po przejściu przez wszystkie kroki użytkownik zobaczy wycenę, swojego zamówienia, oraz będzie mógł wypełnić formularz kontaktowy.

3.2.1 Wybór i edycja pomieszczenia

Poprzez wybór pomieszczenia użytkownik określi otoczenie w jakim przedstawiany będzie produkt. Docelowo otoczenie to powinno jak najbliżej imitować miejsce ich instalacji, pomieszczenia w domu klienta. Nigdy to nie będzie możliwe, ale możliwe będzie stworzenie czegoś co docelowe otoczenie w pewnym stopniu przypomina. Ułatwi to klientowi dobranie dopasowanego do siebie produktu.

Użytkownik będzie mógł wybrać jeden z paru widoków takich jak: kuchnia, salon, jadalnia, sypialnia. Będą to różne pomieszczenia o różnym układzie mebli, oraz innym zastosowaniu, aby pokazać różnorodne zastosowanie dekoracji. Każde z nich będzie posiadać interaktywne ściany, o różnych wymiarach, do których będzie można przymierzać panele ścienne.

Wybrać będzie można styl i kolorystkę pomieszczenia. Każde z nich będzie posiadać zdefiniowany zbiór palet kolorów, które będą zmieniać kolorystykę całego pomieszczenia. Ściany będą zmieniały kolor na inny, podłogi będą miały inną teksturę, a niektóre meble będą miały zmieniony model aby stylistycznie lepiej pasować do wnętrza.

3.2.2 Katalog asortymentu

Widok programu zostanie podzielony horyzontalnie na dwie części. Górna będzie przedstawiać wizualizację, a dolna będzie oknem wyboru opcji gdzie przeglądać i wybierać będzie można katalog asortymentu. Przeciągając w lewo i prawo wybrać będzie można kształt paneli. Do wyboru będą kwadratowe, prostokątne, trójkątne, sześciokątne, oraz inne kształty które nie są wielokątami foremnymi. Podczas przeglądania poszczególnych form, wizualizacja na ścianie będzie się zmieniać, aby lepiej oddać wygląd nowego panelu.

Następnie wybrać będzie można sposób rozmieszczenia paneli. Figury geometryczne można ustawiać na wiele sposobów oraz kombinacji, dlatego każdy kształt będzie posiadać alternatywne możliwości ustawienia. Kwadraty i prostokąty mogą być równo poustawiane w kratę, przesunięte o połowę swojej długości, jak cegły w murze. Prostokąty można ustawiać na więcej różnych sposobów, w przeciwieństwie do kwadratów. Niektóre figury będzie można obrócić, przykładowo kwadraty będzie można obrócić o 45 stopni, a trójkąty o 30, lub 45 stopni.

Domyślnie panele będą ustawione bez szczelin między segmentami. Użytkownik będzie mógł ustalić jakie jednakowe odstępy mają być pomiędzy panelami, tworząc estetyczne marginesy pomiędzy nimi. Wartość tych odstępów będzie dowolna i będzie mogą zostać dokładnie ustalona przez użytkownika, ale nie przekraczająca dwukrotności szerokości panelu.

3.2.3 Widok i rozmieszczenie produktu w scenie

Zależnie od wybranej ściany, dostępna powierzchnia będzie miała różne wymiary, przez co zmieścić będzie można różną liczbę paneli na niej. Określić będzie można różną ilość kolumn i rzędów paneli zależnie od ich kształtu, rozmiaru oraz wybranych wcześniej odstępów pomiędzy nimi. Użytkownik będzie mógł określić ilość wierszy, oraz kolumn tak długo jak panele będą mieścić się na zadanej powierzchni.

3.2.4 Poruszanie kamerą

Przez cały proces konfiguracji produktu, będzie można w określonym stopniu obracać widokiem, aby spojrzeć na panele z różnych perspektyw. Możliwe będzie również przybliżenie widoku w określonym stopniu za pomocą gestów albo za pomocą przycisków.

3.2.5 Zmiana oświetlenia

Ostatnią opcją konfiguracji będzie zmiana pory dnia. Za pomocą suwaka będzie można określić porę dnia, która będzie wpływać na oświetlenie pomieszczenia. Słońce oświetlające pomieszczenie będzie zmieniać swoje położenie, a nocą zapalać się będą światła wewnątrz pomieszczenia.

3.3 Określenie stopnia interaktywności i zmiennych elementów otoczenia

Ważne jest określenie stopnia interaktywności wizualizacji. Określić trzeba elementy interaktywne i zakres ich interaktywności. Pozwoli to na określenie czasu potrzebnego na wykonanie projektu i lepsze zarządzanie projektem.

3.3.1 Zbiór zmiennych elementów otoczenia i stopień ich edycji

Niektóre elementy otoczenia będą zmienne, co wymaga określenia zbioru tych elementów oraz sposobu w jaki będą się one zmieniać. Sposób ich edycji oraz zasoby wymagane do ich implementacji również. Niektóre zmienne parametry, takie jak kolor ścian, będą mogły zostać automatycznie zmienione na dowolny inny kolor za pomocą logiki programu, jednak edycja innych rzeczy, takich jak modele nie mogą być tak łatwo sparametryzowane, oraz wymagane będzie stworzenie osobnych modeli trójwymiarowych na tę okazję.

Dodanie możliwości edycji otoczenia będzie mogło być dodane na parę różnych sposobów zależnie od edytowanego elementu otoczenia. Wydzielić można trzy główne grupy: modele trójwymiarowe, dodatkowa logika programu i implementacje scen. Statyczne przedmioty będą potrzebowały modeli trójwymiarowych. Pomieszczenia, a dokładniej sceny je przedstawiające będą wymagały implementacji, w tym przypadku złożenia ich razem. Ostatnią rzeczą będą dodatkowe efekty, które wymagać będą dodatkowej logiki takie jak zmiana pory dnia, czy zmiana koloru ścian i pewnych elementów.

Modele obiektów będą statyczne oraz nie będzie możliwości edytowania ich modeli. Jedyne co będzie możliwe to podstawowe skalowanie ich oraz podmiana ich modeli trójwymiarowych w locie. Dlatego wymagane będzie stworzenie osobnych modeli trójwymiarowych. Edytując elementy wnętrza podmieniać będziemy całe modele, zmieniać tekstury oraz w pewnych przypadkach zmieniać materiał na taki który będzie bardziej oddawał wymagany fizyczny materiał. Zasoby te będą wymagane do stworzenia pokoju jego elementów, oraz mebli się w nim znajdujących.

Wymagane będzie stworzenie modeli od podstaw, o odpowiednich proporcjach, wymiarach i kroju. Osobne elementy wnętrza będą posiadały zupełne odrębne modele, meble będą posiadały swoje osobne warianty modeli trójwymiarowych,

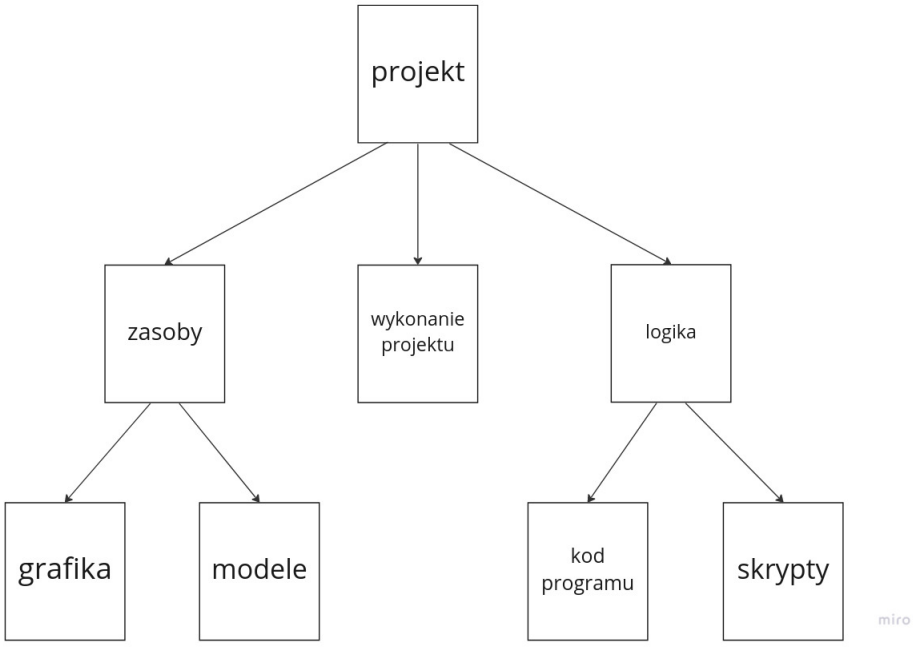
które jedynie w pewnym stopniu różnić będą się od swojej bazowej wersji.

Dodatkowa logika będzie musiała zostać zaimplementowana do odtworzenia zmiany pory dnia. Słońce będzie zmieniać swoje położenie rzucając cień pod różnymi kątami, a nocą zapalać będą się światła wewnątrz pomieszczenia o innym zabarwieniu. Światło słoneczne będzie miało biały kolor, a nocą żółto-pomarańczowy. Będzie musiało to być odpowiednio oprogramowane, oraz dostosowane do każdego pomieszczenia z osobna.

Kolejną rzeczą jest zmiana kolorów ścian i elementów na dowolny wybrany kolor. Dostępna będzie narzędzie do wybierania kolorów, dzięki któremu będzie można dobrać dowolny kolor ścian. Wpływać on będzie na wygląd oraz postrzeganą jasność w pomieszczeniu. Kolorować będzie można również podłogę w pomieszczeniu, będzie ona zachowywać swoją teksturę, ale będzie ona barwiona na podany kolor.

3.4 Podział pracy wymaganej do wykonania projektu

Pracę wykonaną na potrzeby programu można podzielić na trzy oddzielne kategorie. Wykonać należy materiały do których wliczają się elementy artystyczne niezbędne do wykonania projektu, czyli modele trójwymiarowe i grafiki. Wymagana jest logika programu, która będzie definiować w jaki sposób będzie się on zachowywał. Logikę i zachowanie programu będzie definiować kod programu, oraz skrypty napisane na jego potrzeby.



Rys. 1. Podział pracy na różne kategorie

Ostatnią rzeczą jest złożenie tych poszczególnych części w jeden działający projekt w edytorze Unity 3d.

3.4.1 Modele i zasoby graficzne

Na zasoby wymagane do stworzenia projektu składać się będą modele trójwymiarowe oraz grafiki. Modele trójwymiarowe i tekstury zostaną użyte do stworzenia obiektów w wirtualnej scenie. Na grafiki składać będą się zdjęcia, ikony i inne obrazy wymagane w graficznym interfejsie użytkownika. Obie te grupy będą statycznymi zasobami które zostaną użyte w projekcie, a większość z nich trzeba będzie wykonać własnoręcznie.

3.4.2 Modele trójwymiarowe

Głównymi elementami tworzącym scenę będą modele trójwymiarowe. Wszystko co będzie zbudowane w wirtualnej scenie będzie zbudowane z tych modeli trójwymiarowych. Każdy z nich będzie musiał zostać osobno ręcznie wymodelowany na podstawie zdjęć i rysunków poglądowych. Wszystkie z nich będą musiały zachowywać odpowiednie proporcje, względem przedmiotu który odwzorowują. Kolejnym ważnym aspektem jest jednolita skala modeli w projekcie, dzięki temu, nie trzeba będzie ręcznie dopasowywać rozmiaru każdego z modeli z kolei.

Meble w scenie będą mogły zostać użyte jako pojedyncze wolnostojące meble, albo parę modułów mebli stanowiące większą całość, przykładem może być zabudowa kuchni gdzie parę szafek stanowią większą całość. Podobną zasadę można zastosować w większej skali, do zbudowania ścian pomieszczenia. Zamiast modelować wszystkie ściany pomieszczenia jako jeden model, można wymodelować parę różnych wariantów ściany, a potem z nich złożyć każdy dowolny pokój.

Wymodelowane będzie parę wariantów ścian. Oprócz zwykłej ściany, będą również warianty, posiadające miejsce na wstawienie drzwi, oraz wariant z miejscem na okna i parapet. Moduły ścian będą podzielone i zaprojektowane w taki sposób, że będzie można je rozciągać do wymaganej szerokości, tworząc pokój o dowolnych wymiarach i kształtach. Dla urozmaicenia ścian, okna będą wstawione we wnęki, w które opcjonalnie wstawić będzie można parapety. Dodatkową ozdobą ścian będą listwy przy podłodze oraz sztukaterie przy suficie. Oba te elementy będą łatwe w rozciągnięciu i dopasowaniu do ścian ze względu na swoją jednowymiarowość.

Elementem urozmaicającym ściany są też grzejniki, które wstawione są zazwyczaj we wnękę pod parapetem okiennym. Ważne jest to aby również wymodelować rury, które w rzeczywistości doprowadzają ciepłą wodę do grzejnika. Grzejniki żeberkowe będą musiały zostać wykonane w różnych szerokościach gdyż nie można ich łatwo skalować. Bardziej nowoczesne grzejniki posiadającą pofalowaną powierzchnię są bardziej przyjazne skalowaniu modelu.

Meble będą posiadały bardziej skomplikowane i szczegółowe modele trójwymiarowe, oraz posiadać będą swoją teksturę. Meble nie będą mogły być tak łatwo skalowane aby nie zaburzyć ich proporcji. Ważnym aspektem jest nałożenie tekstury, która czasami będzie musiała zostać realistycznie nałożona na model, przykładem mogą być drewniane meble, które tworzone są w bardzo określony sposób. Zaburzenie tego przyzwyczajenia, będzie się podświadomie rzucało użytkownikowi w oczy.

Niektóre meble są stworzone z więcej niż jednego materiału, więc wymagane będzie nałożenie dwóch lub więcej różnych obrazów jako jedną teksturę. W pewnych przypadkach nie będzie to wystarczające, oraz wymagane będzie podzielenie jednego obiektu na więcej niż jeden model trójwymiarowy, żeby móc wykonać go z dwóch, lub więcej, różniących się od siebie materiałów. Przykładem może być okno, którego rama jest wykonana z tworzywa sztucznego, albo drewna, ale jego szyby są szklane, czyli półprzeźroczyste.

Niektóre obiekty sceny, ze względu na użyte różne materiały, będą musiały być podzielone na parę osobnych części. W przypadku okien muszą posiadać one swoje klamki, które są wykonane z innego materiału. Podział jednego obiektu na mniejsze uprości proces modelowania poprzez ponowne użycie wcześniej użytych modeli. W tym przypadku, klamki w każdym wariancie okien, będą takie same, więc można oszczędzić czas używając osobno tych samych klamek dla wszystkich okien.

Wnętrze posiadać będzie również inne elementy, które sprawią że pomieszczenie będzie wyglądać bardziej jak prawdziwy dom. Przewidywane są dywany i niektóre artykuły RTV/AGD. Dywan będzie prostą powierzchnią z teksturą dywanu i nałożoną na mapą wysokości. Ważnym elementem, sprawiające że pomieszczenie będzie wyglądać jak dom, są sprzęty

domowe. Przykładem jest telewizor, który jest często centralną częścią salonu, dlatego również trzeba będzie wymodelować parę różnych modeli telewizora, w różnych rozmiarach.

Kolejnym elementem do wymodelowania są lampy i żyrandole. Na pierwszy rzut oka będą kolejnymi obiektami które będą uwiarygadniać całą scenę, ale ich ukrytą funkcją będzie również oświetlanie sceny, podczas nocy. Okna będą przepuszczały światło słońca za dnia, a w nocy lampy będą częściowo oświetlać scenę, aby można byłoby zobaczyć pokój i jego otoczenie w różnych warunkach oświetleniowych.

3.4.3 Grafiki i obrazy

Na grafiki wymagane na potrzeby projektu składają się tekstury, obrazy użyte w scenie, oraz obrazy i ikony stworzone na potrzeby interfejsu użytkownika.

Tekstury będą obrazami nałożonymi na modele trójwymiarowe. W przypadku bardziej skomplikowanych modeli będą one nałożone na model podczas modelowania, wymagane również może być odpowiednie dopasowanie tekstury do modelu. W przypadku mniej skomplikowanych modeli, tekstura będzie mogła zostać potraktowana jako jednolity kolor, albo powtarzalny materiał, który zostanie stworzony i zaaplikowany do modelu, w silniku do gier komputerowych.

Kolejnym typem grafiki użytym w projekcie będą obrazy i zdjęcia umiejscowione w świecie gry. Będą one wspomagać iluzję prawdziwego, naturalnego otoczenia. Mogą to być tekstury chmur za oknem, albo panoramiczna tekstura, przedstawiająca niebo 360-ciu stopniach widoczna za oknami mieszkania.

Ostatnim typem będą grafiki i ikony użyte bezpośrednio w interfejsie użytkownika. Wcześniejsze sposoby tworzenia interfejsu użytkownika w Unity3D wymagały aby każdy wystylowany przycisk był osobnym obrazem, było to bardzo pracochłonne i wymagało stworzenia osobnego obrazka dla guzików, które różniły się od siebie jedynie napisami. W projekcie użyty będzie nowszy sposób tworzenia interfejsu użytkownika, User Interface Toolkit, który ogranicza użycie obrazów do niezbędnego minimum. Pozwala on na bardziej programowe podejście do tworzenia interfejsu i jego stylowania.

Na potrzeby interfejsu będą potrzebne ikony i miniatury. Ikony użyte będą w niektórych elementach interfejsu jako urozmaicenie, albo w miejscach gdzie słowny opis albo fraza nie zmieści się w elemencie interaktywnym. Takim przykładem będą ikony przedstawiające poszczególne stadia modyfikacji produktu, albo przyciski finalizujące kolejne kroki. W tym celu projekcie użyty zostanie pakiet ikon Font Awesome, który używany jest w wielu stronach internetowych.

W niektórych przypadkach wymagany tekst się nie zmieści, a treść do przekazania będzie zbyt skomplikowana aby przekazać ją za pomocą prostej ikony. W takich przypadkach użyte zostaną miniatury. W przypadku graficznego interfejsu miniatury przedstawiać będą skomplikowane koncepty albo będą małymi obrazami przedstawiającymi paletę kolorów a nie szczegóły zdjęcia. Miniatury będą przedstawiać różne palety kolorów w pomieszczeniu, nie jest ważne czy nawet przedstawiają ten sam pokój a ich kolorystyka. Bardziej prostymi miniaturami będą te przedstawiające kształty paneli, oraz te przedstawiające różne wariacje ich ułożenia.

Wymagane będzie wygenerowanie miniatur przedstawiających palety, będą one zrzutami ekranu pomieszczenia gdzie jego elementy będą posiadały zmienione materiały. Miniatury przedstawiające unikalne sposoby ułożenia paneli oraz ich kształty będą musiały zostać wykonane własnoręcznie.

3.4.4 Logika programu

Gdyby projekt był pisany od podstaw wymagane byłoby napisanie jakieś jego centralnej logiki, w tym przypadku użyty jest silnik do gier komputerowy Unity3D. Dzięki temu podejściu jedyne co trzeba będzie stworzyć to elementy świata gry, interfejs graficzny oraz fragmenty kodu, które będą tę logikę implementować.

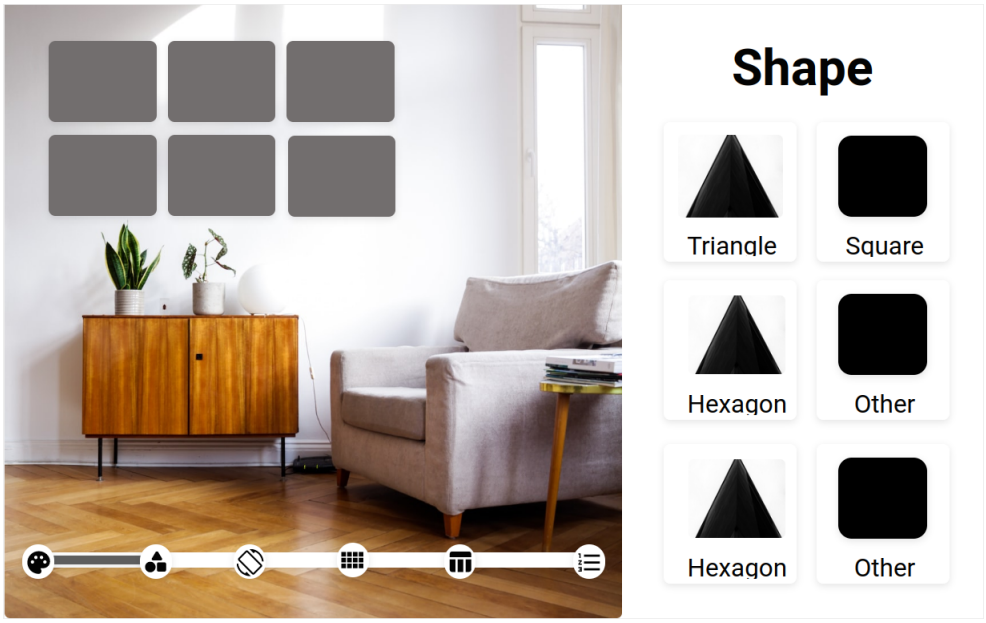
Każdy element w grze tzw. 'GameObject' może posiadać różne komponenty, to właśnie one definiują funkcje i rolę tego obiektu w grze. Dodanie komponentu z określonym skryptem dodawać będzie pewną funkcjonalność do danego obiektu w scenie. Obiekty których materiały będą się zmieniać, wedle palety, będą posiadać określony skrypt jako swój komponent. Kamera zmieniająca swoje położenie i ustawienie, również będzie posiadać od tego osobny skrypt. [11]

Poprzez graficzny interfejs użytkownika klient będzie wprowadzał kolejne informacje, oraz wchodził w interakcję z programem. Dlatego część logiki będzie podpięta pod obiekty interfejsu użytkownika, odpowiedzialne one będą za aktualizowanie potencjalnego zamówienia, oraz uruchamianie akcji wyzwalanych po interakcji z elementami z interfejsem. Dzięki nim możliwe będzie przechodzenie przez kolejne kroki konfigurowania produktu.

3.4.5 Interfejs użytkownika

Zadaniem interfejsu użytkownika będzie umożliwienie mu wejścia w interakcję z programem. Dzięki niemu możliwe będzie przeprowadzenie go przez kolejne kroki konfiguracji produktu, dostrajanie kolejnych jego parametrów, wyświetlenie ekranu podsumowania i wypełnienie formularza kontaktowego.

Dostosowanie produktu



Rys. 2. Wstępny zarys interfejsu użytkownika

Pierwszym układem będzie układ przedstawiający produkt w scenie, umożliwiający dostosowanie produktu. Dzielić się on będzie na dwie części. Po lewej widoczna będzie trójwymiarowa wizualizacja, w której widoczne będą panele dekoracyjne. Wszystkie zmiany wprowadzane przez użytkownika będą aktualizowały wizualizację. Ekran wizualizacji będzie pozwalał na ograniczone obracanie kamerą i przybliżanie i oddalanie jej. Możliwe też będzie przełączanie się

pomiędzy różnymi ścianami w pomieszczeniu.

Na dole widoku wizualizacji widać oś przedstawiającą kolejne etapy konfiguracji produktu. Poszczególne ikony będą przedstawiać kolejne kroki jego konfiguracji. Oś ta będzie pełniła funkcję paska postępu, informując użytkownika ile jeszcze pracy musi on wykonać aby zobaczyć efekt końcowy. Dzięki niej będzie możliwe przechodzenie w przód i w tył pomiędzy etapami, poprzez kliknięcie ikony symbolizującej dany etap.

Część ekranu po prawej będzie obszarem przeznaczonym na przyciski, przełączniki i inne elementy interaktywne. Cała jego zawartość i układ będzie się zmieniać zależnie od aktualnego etapu konfiguracji i pozwalać będzie na dostrojenie jednego z aspektów produktu. Na górze każdego z nich będzie nazwa etapu, która jednoznacznie wytłumaczy co dokładnie zmieniać w produkcie będzie dany parametr. Wprowadzanie danych będzie odbywać się poprzez przyciski, suwaki i pola wyboru do zaznaczenia.

Etap wyboru palety kolorów będzie posiadał przyciski z miniaturami pomieszczeń, każdy z nich będzie posiadał swoją własną nazwę widoczną pod obrazkiem. Kliknięcie w miniaturkę zmieni kolorystkę wnętrza.

Kolejny etap, wyboru kształtu paneli posiadać będzie zbiór możliwych do wybrania kształtów paneli dekoracyjnych. Wszystkie panele będą jednakowe, dlatego wybór ich kształtów będzie dosyć ograniczony. Po kliknięciu, modele paneli będą zmienione na te o innym, wybranym kształcie.

Potem możliwe będzie obrócenie paneli z puli zdefiniowanych wartości obrotu, dla każdego kształtu warianty obrotu będą inne gdyż różne kształty posiadają różną ilość osi symetrii. Wybranie opcji sprawi, że wszystkie panele zmienią swoje ułożenie oraz obrócą się.

Kolejne będzie ustawienie szczelin pomiędzy panelami, osobno będzie można ustawić wartość horyzontalnych jak i wertykalnych szczelin pomiędzy panelami. Odbywać się to będzie z użyciem suwaków. Będą one miały z góry określone granice. Każda edycja suwaka zaktualizuje szerokość szczelin pomiędzy panelami.

Zależnie od wielkości pustej ściany możliwe będzie zmieszczenie różnej liczby rzędów i kolumn paneli. Możliwe będzie zwiększenie albo zmniejszenie liczby kolumn i rzędów paneli, które składać będą się w jedną całość. Maksymalna ich liczba będzie zależna od wielkości ściany, kształtu, obrócenia i sposobu ułożenia paneli. Po aktualizacji liczby paneli, będą one ustawiały się w centrum wybranej ściany.

Widok podsumowania

Kolejnym osobnym widokiem następującym po zakończeniu procesu konfiguracji jest widok podsumowania. Będzie w nim widoczna lista paneli, które będą wymagane do zrealizowania zamówienia. Wypisana będzie ich liczba, nazwa, dokładny kolor, oraz cena za sztukę. Pod linią symbolizującą podsumowanie widoczna będzie łączny koszt wszystkich paneli. Po kliknięciu przycisku Będzie można przejść do widoku formularza kontaktowego.

Formularz kontaktowy

Ostatnim widokiem będzie formularz kontaktowy. Użytkownik wprowadzi w nim swoje dane kontaktowe, aby firma mogła się z nim skontaktować w celu potwierdzenia realizacji zamówienia. Po kliknięciu na przycisk 'Wyślij' zamówienie powinno być dodane do systemu a dane klienta wysłane na serwery firmy.

3.4.6 Skrypty

Zadaniem skryptów będzie urozmaicenie martwej i nieruchomej sceny wirtualnego pomieszczenia, dzięki nim będzie można zmieniać wygląd pomieszczenia i paneli, oglądać panele z różnych stron i zmieniać oświetlenie w pomieszczeniu.

Cała logika programu będzie zaimplementowana za pomocą luźno połączonych ze sobą fragmentów kodu, napisanego w języku C#. Każdy z nich będzie miał inne zadanie, oraz zostanie przyłączony do odpowiadającego jemu elementowi wirtualnej sceny, pozwoli to na zachowanie porządku i spójności w projekcie. Każdy z nich będzie realizował jedną z wielu interaktywnych funkcji programu.

Najważniejszym skryptem będzie skrypt układający panele na ścianie. Ściana interaktywna będzie płaską powierzchnią, do której dodany zostanie skrypt ją implementujący. Na płaską powierzchnię o danych wymiarach nakładane będą w różnych konfiguracjach, panele w trzech podstawowych kształtach: trójkątnym, kwadratowym i sześciokątnym. Maksymalna liczka rzędów i kolumn będzie obliczana na podstawie wymiarów indywidualnej ściany, dzięki czemu różne interaktywne ściany będą mogły mieć różne wymiary. Podczas jej obliczania będą brane pod uwagę rzeczywiste wymiary ściany interaktywnej, wymiary poszczególnych paneli, oraz marginesy pomiędzy nimi.

Kolejnym skryptem będzie skrypt odpowiedzialny za poruszanie kamerą, będzie on dołączony bezpośrednio do obiektu kamery. Będzie on definiował sposób obracania się kamery, oraz jej zachowanie.Kamera będzie zawsze zwrócona w kierunku ściany oraz poruszać się będzie względem ściany, gdzie punktem przyłożenia będzie ściana. Maksymalne kąty wychylenia w każdą ze stron będą ręcznie zdefiniowane dla każdej ze ścian osobno.

Osobny skrypt będzie on definiował położenie początkowe kamery, oraz możliwe wychylenia kamery. Skrypt ten będzie dołączany do każdej interaktywnej ściany z kolei. Posiadać on będzie informacje na temat możliwego wychylenia kamery, w każdym kierunku osobno, oraz jaka będzie jej minimalna i maksymalna odległość kamery od interaktywnej ściany.

Zmiana palety pomieszczenie będzie realizowana poprzez dwa różne rodzaje skryptów. Pierwszy będzie służył do zarządzania globalną paletą kolorów, a drugi typ skryptu dołączany do każdego elementu, którego materiały będą się zmieniać zależnie od palety kolorów pomieszczenia.

Kontroler palety kolorów dołączony będzie do któregoś z głównych obiektów sceny. Będzie on wyszukiwał wszystkie instancje skryptu trzymającego paletę kolorów oraz będzie inicjować funkcję zmiany materiałów.

Będzie on reprezentował listę materiałów z których wykonany może zostać obiekt, oraz posiadać będzie publiczną funkcję pozwalającą na ustawienie materiału o danym indeksie. Dodatkowo możliwe będzie pokolorowanie danego materiału, co będzie bardzo pomocne w zmianie koloru paneli dekoracyjnych. Paleta kolorów każdego z elementów otoczenie będzie inna, oraz będzie ręcznie stworzona ze stworzonych już materiałów.

W podobny sposób będzie zrealizowane oświetlenie wirtualnej sceny. Główny kontroler odpowiadający za oświetlenie będzie zarządzał trzema różnymi rodzajami oświetlenia. Pierwsze będzie oświetlenie ogólne, które świeci w jednym kierunku, oraz nie tworzy cieni i stanowić będzie większość oświetlenia sceny. Możliwa będzie zmiana barwy światła na chłodniejszy, lub cieplejszy aby lepiej oddać porę dnia.

Drugim rodzajem światła, będzie światło słoneczne. Stworzony będzie obiekt imitujący słońce, świecący chłodnym w swojej barwie światłem, ten typ światła będzie rzucał realistyczne cienie w pomieszczeniu wpadające przez okna.

Ostatnim typem światła będą obiekty, które będą lampami w pomieszczeniu. W każdej lampie będzie umieszczone osobne źródło światła, oraz skrypt pozwalający na kontrolowanie go. Kontroler oświetlenia będzie wyszukiwał wszystkie skrypty lamp oraz zmieniał ich ustawienie wedle potrzeb, traktując wszystkie lampy jako jedno grupowe oświetlenie.

3.4.7 Wykonanie projektu

Na wykonanie projektu składać będą się akcje wykonane głównie w edytorze Unity3D, polegać one będą głównie na składaniu stworzonych zasobów ze sobą w jeden działający projekt. Jedną z pierwszych rzeczy, które trzeba będzie zrobić to stworzenie wirtualnego pomieszczenia, które będzie częścią sceny gry.

Pomieszczenie ma za zadanie stworzyć środowisko, w którym można będzie pokazać panele dekoracyjne z zachowaniem ich skali, oraz zastosowania. Nie musi ono być superrealistyczne, ale być na tyle wiarygodne żeby stanowić dobrą refernecję dla przedstawianego produktu.

4 Realizacja aplikacji konfiguratora wnętrz z panelami dekoracyjnymi

Po rozplanowaniu projektu można przejść do jego realizacji. Przed przystąpieniem do zaprogramowania logiki, wymagane będzie stworzenie grafki i modeli, które tworzyć będą wirtualną scenę. Żeby wirtualne pomieszczenie było wiarygodne, musi ono być dokładnym odwzorowaniem istniejącego pomieszczenia.

4.1 Rozplanowanie scen i pomieszczeń

Jednym z głównych zadań wizualizacji jest prezentacja produktu w naturalnym dla niego otoczeniu, dlatego ważne jest aby zaprojektować pomieszczenia, które będą wyglądać lepiej dzięki powieszeniu paneli na ścianach. Celem tych paneli jest ozdobienie i przełamanie monotonii dużych pustych ścian danego pomieszczenia, dlatego prezentowane pomieszczenie powinno posiadać dużą pustą ścianę. Ściana nie musi być całkowicie wyeksponowana, przy niej mogą stać meble, ale nie powinny one być zbyt wysokie, dlatego panele pasowałyby również nad łóżkiem w sypialni. Pomieszczenie musi posiadać dużą nieprzerwaną ścianę, zazwyczaj salon jest największym pokojem w mieszkaniu, ale korytarze też mogą posiadać długie i monotonne powierzchnie.

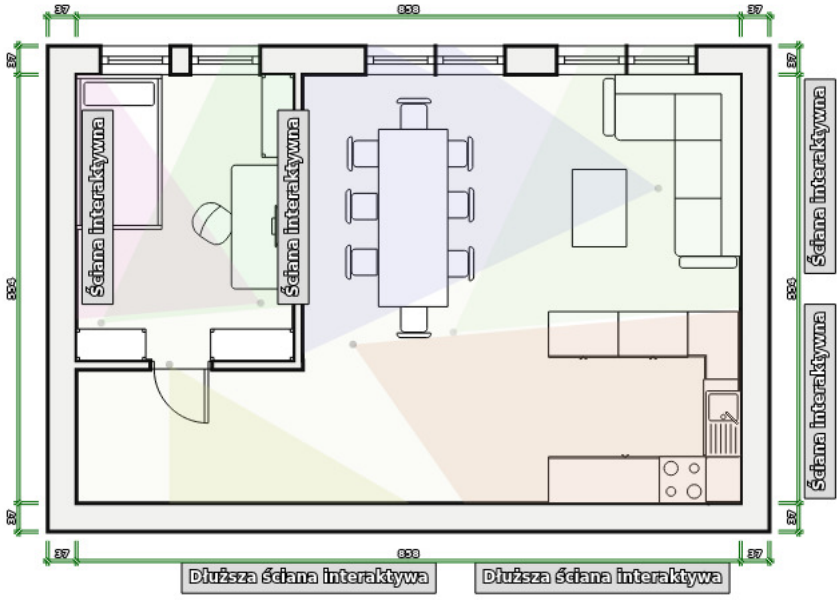
Zazwyczaj na zdjęciach wizualizacjach pokazany jest jeden wąski kadr przedstawiający produkt w swoim otoczeniu. Można by lepiej użyć medium interaktywne poprzez wyznaczenie paru interaktywnych ścian w jednym pomieszczeniu lub mieszkaniu, pomiędzy którymi można by się przełączać. Każda z nich byłaby w trochę innym otoczeniu, symbolizujące inne pomieszczenie. Kamera zamiast przeskakiwać do osobnego pomieszczenia przesuwałaby się płynnie na kolejne miejsce i obracała kadr w kierunku nowej interaktywnej ściany. Wymaga to zaprojektowania mieszkania specjalnie pod tym kątem.

Zaprojektowanie pomieszczeń pozwoli na określenie ilości i umiejscowienia interaktywnych ścian pomieszczenia, rozplanowanie rozmieszczeń kamery oraz jej kadrowanie. Ważnym aspektem jest też rozplanowanie naturalnego i sztucznego oświetlenia w pomieszczeniu co odbywa się poprzez rozmieszczenie okien, lamp i żyrandoli. Po wstępnym zaprojektowaniu pomieszczeń, wiadoma będzie liczba i rodzaj mebli, które będą potrzebne do jego odtworzenia i umeblowania pomieszczenia za pomocą modeli trójwymiarowych. Dopiero wtedy można przejść do kolejnego punktu, którym jest wykonanie listy wymaganych modeli, oraz wymodelowanie ich. Dodatkowo, gotowy plan pomieszczenia ułatwi odwzorowanie go w silniku Unity3D.

4.1.1 Plan pomieszczenia

Poniżej przedstawiony jest plan pomieszczenia, które ma znaleźć się w projekcie. Rozplanowanie pomieszczenia pozwoli na określenie miejsc w do których będzie przenosić się kamera, oraz jaki kadr będzie ona obejmować. Na rysunku punkty oznaczają miejsca ustawienia wirtualnej kamery, a kolorowy trójkąt oznacza pole widzenia kamery, które na rysunku wynosi 60 stopni, co jest domyślnym ustawieniem kamery w Unity3D

Pomieszczenie z nich jest fragmentem większego mieszania z aneksem kuchennym i osobnym pokojem. Dzięki jednej dużej przestrzeni w planie mieszkania, jedno pomieszczenie jest w stanie pokazać produkt w paru różniących się od siebie otoczeniach, w salonie, jadalni, kuchni i korytarzu.



Rys. 3. Układ pomieszczenia

4.2 Modelowanie elementów 3d

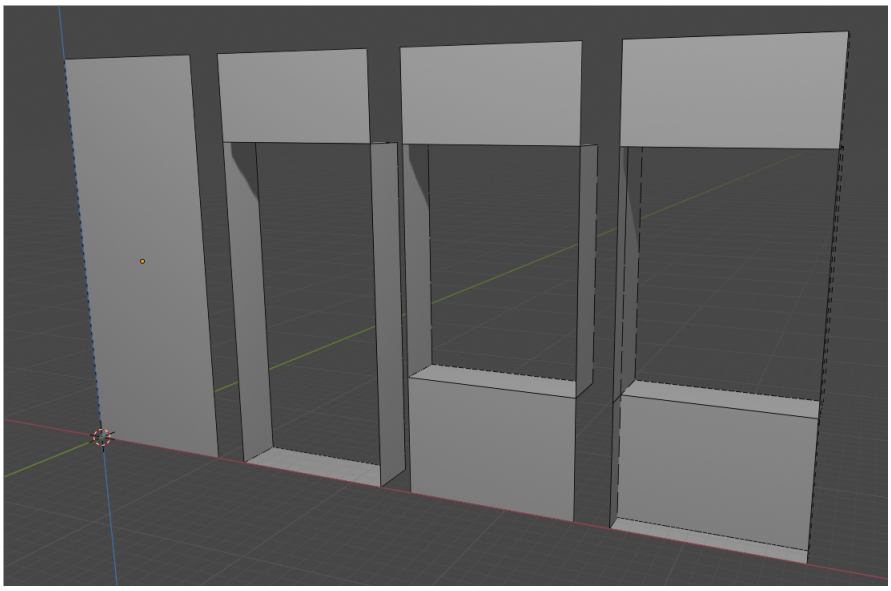
Proste kształty i modele trójwymiarowe można modelować bez technicznych rysunków i planów, takimi modelami będą ściany, które będą prostymi powierzchniami, z opcjonalnymi wnękami i otworami na wstawienie okien lub drzwi. Jedyną rzeczą o której trzeba pamiętać jest zachowanie jednakowej skali oraz proporcji. Podłoga i sufit będzie jeszcze prostsza do wymodelowania, gdyż będzie ona jedynie płaską powierzchnią. Posiadać ona będzie teksturę która urozmaici monotonną powierzchnię. Bardziej wymagające do wymodelowania będą meble, które posiadają bardziej skomplikowane kształty, teksturę oraz materiały.

4.2.1 Modelowanie ścian

Pierwszymi elementami do wymodelowania będą części składające się na pomieszczenie: ściany, podłogi i sufity. Należy je zaprojektować i wymodelować w taki sposób aby możliwe byłoby proste ich rozciąganie, aby zmniejszyć ilość modeli użytych w projekcie. Dodatkowo wydzielić można parę gotowych segmentów, z których można będzie zbudować większość pomieszczeń, będą nimi: pusta ściana, ściana z otworem na drzwi, ścina z otworem na okno, oraz ściana z wnęką pod grzejnik i otworem na okno. Wbrew pozorom z tych gotowych części będzie można złożyć większość podstawowych pomieszczeń. Ważne aby wszystkie modele w projekcie były wymodelowane w tej samej skali, a wymodelowane segmenty ścian powinny przestrzegać podstawowych zasad i wymiarów, którymi kierują się architekci podczas projektowania budynków. Stworzenie niewymiarowych ścian burzyło by immersję użytkownika.

Moduły ścian będą najprostszymi do wymodelowania częściami ze względu na swoją prostą budowę. Będą one miały jednakowe wymiary: szerokość 1-go metra i wysokość 2,5 metra. Pusta ściana będzie jednolitą pionową powierzchnią o określonych wymiarach. Wariant z otworem na drzwi, posiadać będzie fragment ściany nad drzwiami, oraz ściany w które wstawiać będzie się drzwi. Ściana z otworem na okna będzie posiadać wariant uproszczony jedynie z otworem na okno na odpowiedniej wysokości, albo posiadać będzie również wnękę na grzejnik, pod otworem na wstawienie okna.

Przed wyeksportowaniem modelu trzeba pamiętać o pewnych mniej oczywistych zawiłościach, jedną z nich jest



Rys. 4. Modele segmentów ścian

sposób w jaki różne silniki graficzne interpretują modele. Domyślnie w programie blender każda ściana ma dwie strony, tą która jest widoczna, oraz druga która jest całkowicie nie widoczna, pozwala to na nie renderowanie niewidocznych ścian w projekcie. Program blender zawsze pokazuje każdą ścianę z obu stron, ale nie jest to prawdą dla silnika Unity, przez co niektóre ściany modelu mogą być nie widoczne z pewnych stron. Możliwe jest włączenie specjalnego trybu pokazującego które ściany będą widoczne oraz ustawienie ich w taki sposób, aby ściany były widoczne z odpowiedniej strony.

4.2.2 Modelowanie listw ściennych

Ozdobnymi częściami ścian będą listwy przypodłogowe oraz listwy przysufitowe. Oba przykłady listw mają dosyć proste do wymodelowania kształty. Są one jedynie przekrojem rozciągniętym w swojej długości. Dokładny kształt przekroju listwy jest często przedstawiany w katalogu produktów, z uwagi na prostotę modelu jest to wystarczająco dokładna referencja. Z katalogu więc można skopiować przekrój listwy oraz wkleić go bezpośrednio do programu do modelowania jako zdjęcie referencyjne.

Następnie trzeba zeskalować zdjęcie referencyjne do wymiarów zbliżonych do wynikowego modelu, w tej sytuacji bardzo pomaga prawidłowe ustawienie jednostek, oraz siatki w programie Blender. Dopomóc sobie można również ustawiając w pobliżu obiekt o ustalonych wymiarach, na przykład, sześcian o boku 10-ciu centymetrów. Skala i wymiary listwy i tak będą dostosowane podczas modelowania, aby wszystkie listwy miały podobne wymiary i tę samą skalę.

Najpierw trzeba ustawić odpowiednio punkt przyłożenia modelu, aby jego ustawienie było jednakowe z ustawieniem go w modelach ścian. Następnie można wymodelować przekrój listwy, odrysowywując kolejne punkty wedle rysunku poglądowego. Powstały zarys przekroju należy potem rozciągnąć go do końcowej długości listwy. W tym przypadku będzie to standardowa długość dla modułów ścian czyli 1. metr szerokości. Każda listwa posiadać będzie dwa warianty modelu, listwy przycięte prostopadle, oraz pod kątem 45-ciu stopni. Pozwoli to na tworzenie kątów i krawędzi ścian z idealnie schodzącymi się listwami.

W taki sam sposób modelowane są listwy podsufitowe czasami nazywane sztukateriami. Różnicami będzie prze-

sunięty punkt centralny modelu gdyż siatka będzie miała 1 metr a sufit będzie ustawiony na wysokości 2,5 metra. Kolejną różnicą będą dostępne wersje kolorystyczne z uwagi na różnice w materiałach zazwyczaj używanych do ich wykonania. Listwy przypodłogowe wykonywane są z drewna albo plastiku i mają albo teksturę drewna, albo jednolity kolor. Drewniane sztukaterie nie są już spotykane we współczesnych wnętrzach, ale były wykorzystywane w dawnych czasach kiedy ściany również zdobiono drewnianymi panelami. Dlatego sztukaterie nie będą posiadać tekstury, lecz jednolity zazwyczaj jasny kolor.

4.2.3 Modelowanie okien

Okna nie będą tak łatwo skalowalne jak ściany, dlatego będą wykonane w standardowym wariancie pasującym do otworów na nie. By nie używać wszędzie tego samego modelu, trzeba będzie wykonać ich parę wariantów. Dostępne będzie pojedyncze okno, podwójne okno, oraz wysokie okno, które rozmiarami i zastosowaniem przypomina przeszklone drzwi. Rama okna jest dosyć prosta do wymodelowania i można ją wymodelować na podstawie ogólnych wymiarów i zdjęć referencyjnych. Do wymodelowania klamki można użyć rysunków technicznych dostępnych na stronie producenta klamek do okien.

Okno jest pierwszym modelem, który posiadać będzie więcej niż jeden materiał, lub teksturę. Podział jest dosyć prosty, gdzie osobne elementy okna wykonane są z innych materiałów. Rama okna jest wykonana z drewna, albo materiału o podobnym kolorze, szyba będzie przeźroczystą powierzchnią, a cała klamka wykonane będzie z metalicznego materiału o szorstkim wykończeniu.

Niestety nie jest możliwe bezpośrednie wyeksportowanie zaawansowanych materiałów z programu blender do silnika Unity3D. Wyeksportować można jedynie tekstury, albo tekstury powstałe z tychże materiałów. Mimo tego, przydatne będzie na tym etapie przypisanie różnych materiałów do różnych części tego samego modelu, materiały będzie można zastąpić tymi pochodzącymi z silnika Unity3D. Na potrzeby tego projektu materiały silnika Unity3D są wystarczające.

4.2.4 Modelowanie mebli

Modelowanie mebli jest o wiele bardziej skomplikowane ze względu na ich bardziej złożone kształty. Jest jednak pewna metoda, która pozwala na proste i szybkie modelowanie modeli, z użyciem rysunków technicznych modelowanego obiektu. Polega ona na ustawieniu rysunków w przestrzeni trójwymiarowej oraz modelowaniu kolejnych części patrząc jednocześnie na więcej niż jeden rzut. W ten sposób stawiając punkt można sprawdzać poprawność jego umiejscowienia wielu osiach jednocześnie.

Najpierw trzeba znaleźć rysunki techniczne, które przedstawiają modelowany obiekt z przodu, z boku oraz z góry. Czasami rysowane są dodatkowe rzuty, ale w większości przypadków nie są one niezbędne. Ważne jest, aby obiekt w każdym z narysowanych rzutów był narysowany w tej samej skali. By można było je ze sobą zestawić w przestrzeni trójwymiarowej.

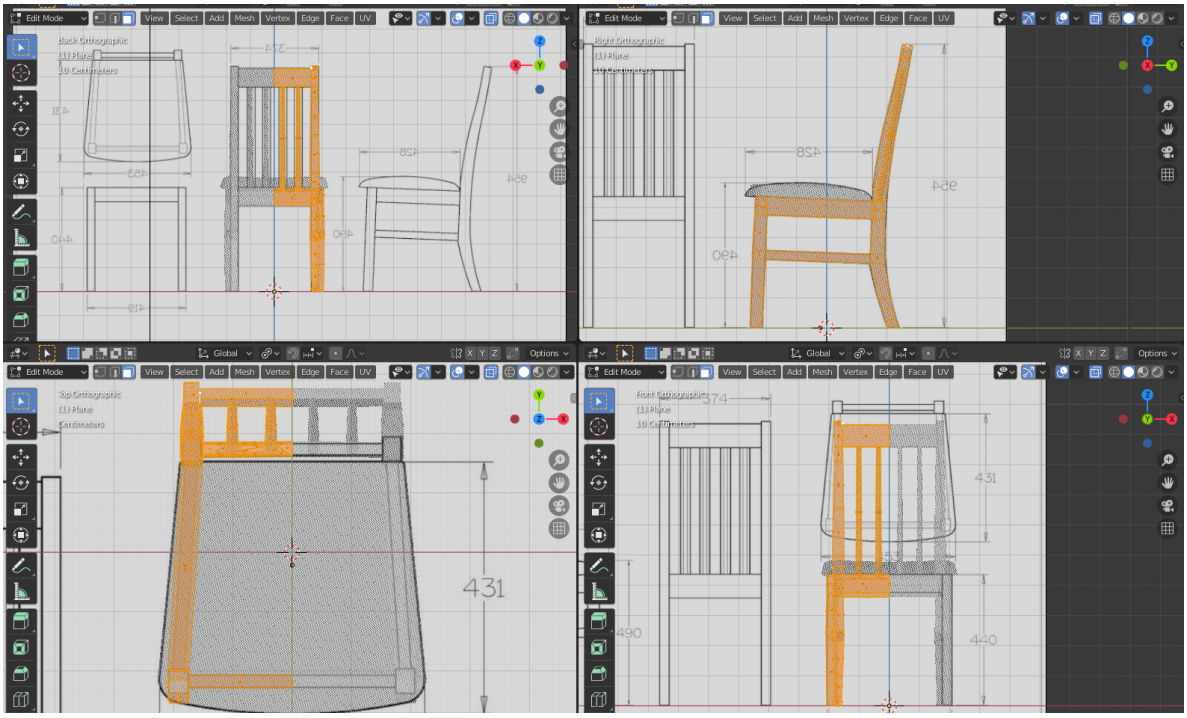
Niektóre firmy meblowe, w ramach marketingu i promocji swoich mebli, udostępniają za darmo modele trójwymiarowe swoich mebli, zdjęcia oraz ich rysunki techniczne. Projekt tworzony na potrzeby tej pracy inżynierskiej nie będzie wykorzystywany komercyjnie, więc można użyć w nim własnoręcznie wykonanych modeli opierających się na rysunkach udostępnionych do publicznego użytku. Czasami nawet udostępniane są gotowe oteksturowane modele trójwymiarowe mebli z zastrzeżeniem nie używania go w projektach komercyjnych. Jedyną różnicą będzie fakt wykonania tychże modeli od podstaw.

Proces modelowania skomplikowanych modeli, zaczynamy od zaimportowania jego rysunków technicznych do programu Blender. Kopiujemy go tyle razy, ile rzutów jest narysowanych na rysunku. Każdy z rysunków ustawiamy w taki sposób aby był widoczny z innej strony: front, bok, góra. Potem tworzymy podstawowy obiekt, kostkę która pomoże zestawić je ze sobą w przestrzeni trójwymiarowej. Kostkę skalujemy do odpowiednich wymiarów, zazwyczaj są to wymiary zakreślające cały obrys przedmiotu, z każdej widocznej perspektywy. Przesuwamy pojedynczo rysunki w jednej płaszczyźnie jednocześnie, aż do czasu kiedy przedmiot w każdym rzucie, mieści się w obrębie kostki.

Kolejnym ułatwieniem medium cyfrowego jest to, że nie jeżeli modelowany przedmiot jest symetryczny, to nie trzeba modelować go w całości. Im więcej osi symetrii ma model tym mniejszą część należy modelować. Krzesło mając jedną oś symetrii można podzielić na dwie połowy, stół natomiast posiada dwie osie symetrii, więc wystarczy wymodelować jego ćwiartkę. W przypadku krzesła wystarczy wymodelować jego połowę, a potem użyć modyfikatora, który stworzy drugą połowę modelu w odbiciu lustrzanym, pamiętać należy, że odbicie to tworzone jest względem punktu centralnego modelu.

Modelowanie krzesła

Teraz można zacząć proces modelowania. Zacząć można od jednego z podstawowych kształtów i powoli dodawać kolejne punkty, linie i ściany. Wielkim ułatwieniem jest fakt, że stawiając nawet jeden kolejny punkt, można sprawdzić poprawność jego położenia w paru rzutach jednocześnie, zamiast przełączać się po kolei pomiędzy nimi.



Rys. 5. Zrzut z ekranu z programu Blender podczas modelowania krzesła

Rama krzesła modelowana jest najpierw jako dwuwymiarowy kształt. Najpierw tworzony jest przekrój nogi krzesła, potem odchodzące od niego poprzeczki, następnie można rozciągnąć dwuwymiarowy obiekt do trójwymiarowego tworząc zalążek ramy siedziska.

Zawczasu w miejscach zamontowania nóg krzesła powstawiane zostały kwadraty, będące jednocześnie przekrojami

każdej z nóg krzesła. Kwadraty będą podstawą do stworzenia kwadratowych nóg krzesła. Tworzenie długich kształtów jest dosyć proste, wystarczy zaznaczyć ścianę która jest spodem nogi, oraz rozciągnąć ją do wybranej długości. Nogę rozciągamy do momentu montowania poprzeczki pomiędzy tylną a przednią noga krzesła. Podczas dostawiania każdego kolejnego przedłużenia na osobnym widoku sprawdzamy jego długość, a na innym jego dokładne położenie w innych osiach. W podobny sposób modelujemy oparcie krzesła, będące przedłużeniem tylnych nóg. Wystarczy powstawiać proste okrągłe szczebelki kończąc krzesło.

4.2.5 Tekstury i materiały dla modeli

Na każdy obiekt w scenie składa się z trójwymiarowego modelu oraz graficznej warstwa w którą jest on pokryty. Obrazy, które nakładane są na te modele nazywane są teksturami. Jeżeli materiał z którego jest dany przedmiot jest jednolity, można zamiast tekstury użyć materiału, w innych przypadkach wymagane jest odpowiednie rozwinięcie modelu oraz dokładne nałożenie tekstury z uwzględnieniem umiejscowienia jej w modelu.

Zadaniem tekstur jest sprawienie aby modele wyglądały bardziej podobnie do swojego oryginału ze świata rzeczywistego. Niektóre obiekty posiadają jednolity kolor, ale niektóre posiadają charakterystyczną, nie powtarzającą się kolorystykę, w takich przypadkach wymagane jest użycie tekstur. Przykładem mogą być meble wykonane z drewna, gdzie widać słoje drewna, albo elementy wnętrza wykonane z kamienia takiego jak marmur.

Tekstury można stworzyć w różny sposób, można stworzyć ją od podstaw za pomocą programów graficznych, albo za pomocą specjalnych programów do tworzenia i generowania wirtualnych materiałów, na przykład Material Maker. Innym podejściem jest użycie zdjęć do ich stworzenia. Na potrzeby projektu, nie będziemy tworzyć tekstur od podstaw, lecz użyjemy już tych istniejących oraz dostępnych w internecie. Ważne jest aby ich licencja pozwalała na nieodpłatne używanie ich w naszym projekcie.

W internecie można znaleźć parę stron internetowych z teksturami i materiałami, ważna jest licencja pod którą podlegają. My będziemy używać materiałów objętych licencją Creative Commons CC0 License, która określa materiał jako podany do domeny publicznej, która pozwala na kopiowanie, modyfikowanie, dystrybucję a nawet na używanie ich w produktach komercyjnych bez pytania o zgodę ich autora. Wedle oficjalne strony Creative Commons, twory opatrzone licencją CC0 przekazane są do domeny publicznej, a ich autor całkowicie zrzeka się praw autorskich do tego tworu. [1]

Do wyboru jest wiele najróżniejszych materiałów, ale trzeba pamiętać aby ograniczyć ich liczbę do minimum z uwagi na ich rozmiar. Jeżeli oddamy wbyt wiele tekstur i plików, problemem może się stać ograniczone miejsce pamięci RAM na urządzeniu odtwarzającym projekt. Dlatego należy wybrać jak najmniej, najbardziej różniących się od siebie materiałów. Kolejnym sposobem na oszczędzenie pamięci jest użycie białych wariantów tekstur, które można na poczekaniu pokolorować aby uzyskać iluzję zupełnie innego, nowego materiału, zamiast przechowywać parę różnych wariantów różniących się jedynie kolorami.

Ogrom materiałów do wyboru może przestraszyć ale można szybko zawęzić zakres poszukiwań wypisując nazwy materiałów, z których zrobione są meble oraz wnętrze. Na większość mebli składa się głównie metal, drewno oraz obicia z tkaniny. Są również inne materiały, takie jak papier, plastik i różne rodzaje kammiennych powierzchni.

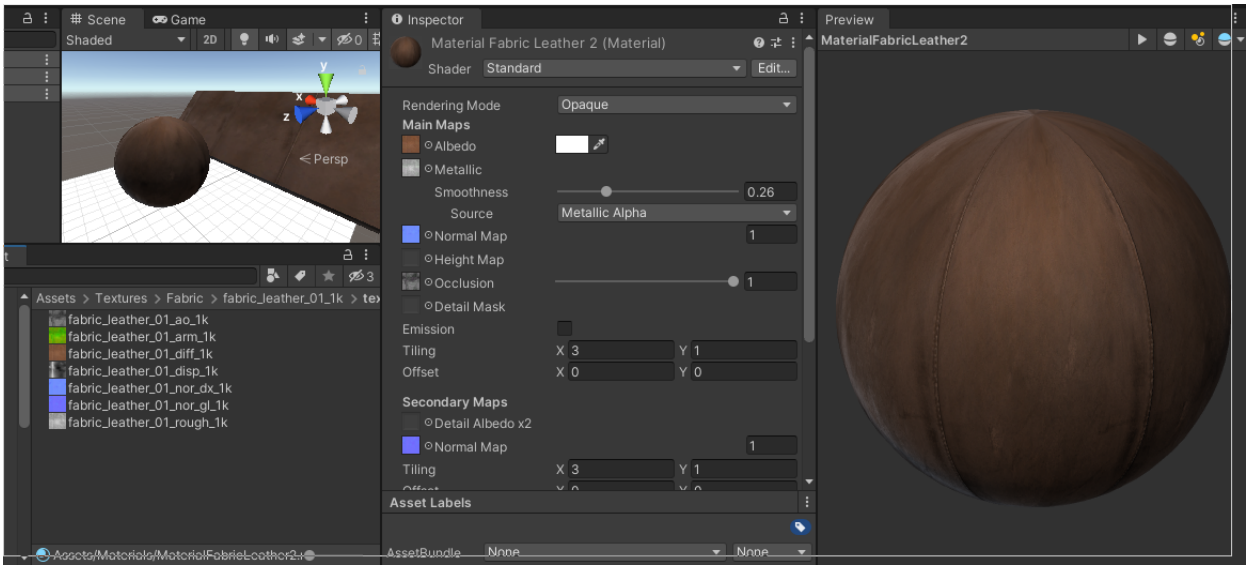
4.2.6 Tworzenie materiałów dla mebli

Na potrzeby projektu wymagane będzie stworzenie materiałów przypisywanych modelom, będą to materiały stworzone w silniku Unity3D. Jeżeli mamy już potrzebne materiały, jest to prosty proces. W archiwum z teksturami znajdują

się nie tylko tekstura, ale również inne obrazy reprezentujące warstwy materiału. Tekstura jest zdjęciem lub obrazem imitujące kolor i przerbarwienia materiału. Czasami jest podstawowa część materiału.

Kolejną częścią są mapy wypukłości, są one specjalnym obrazem, przedstawiającym ostre krawędzie i nierówności w teksturze. Interpretowana jest ona w specjalny sposób przez silnik graficzny, zmienia się ona pod różnymi kątami padania światła. Nawet jeżeli nałożymy mapę wysokości na całkowicie płaską powierzchnię, mapa wypukłości będzie zmieniać się wraz z kierunkiem oświetlenia dając iluzję wypukłości materiału.

Mapa wysokości jest czarno-białym obrazem, w którym jasność piksela odpowiada jego wysokości. Dzięki temu materiał może posiadać iluzję różnych stopni głębokości, nawet jeżeli obiekt jest całkowicie płaski. W projekcie niektóre materiały posiadają osobny obraz odpowiadający okluzji otoczenia (ambient oclusion). Jest to sposób na imitowanie uproszczonego cieniowania obiektów w scenie.



Rys. 6. Proces tworzenia materiału w silniku Unity3D

Dzięki złożeniu tych wszystkich obrazów, możliwe jest odtworzenie wielu materiałów. Wystarczy stworzyć nowy materiał oraz przeciągnąć odpowiednią teksturę do odpowiadających im pól.

4.3 Projektowanie i programowanie logiki programu

Po stworzeniu wszystkich statycznych elementów, można zacząć składać projekt w całość oraz dodać do niego logikę, która sprawi, że możliwe będzie jego użycie i działanie.

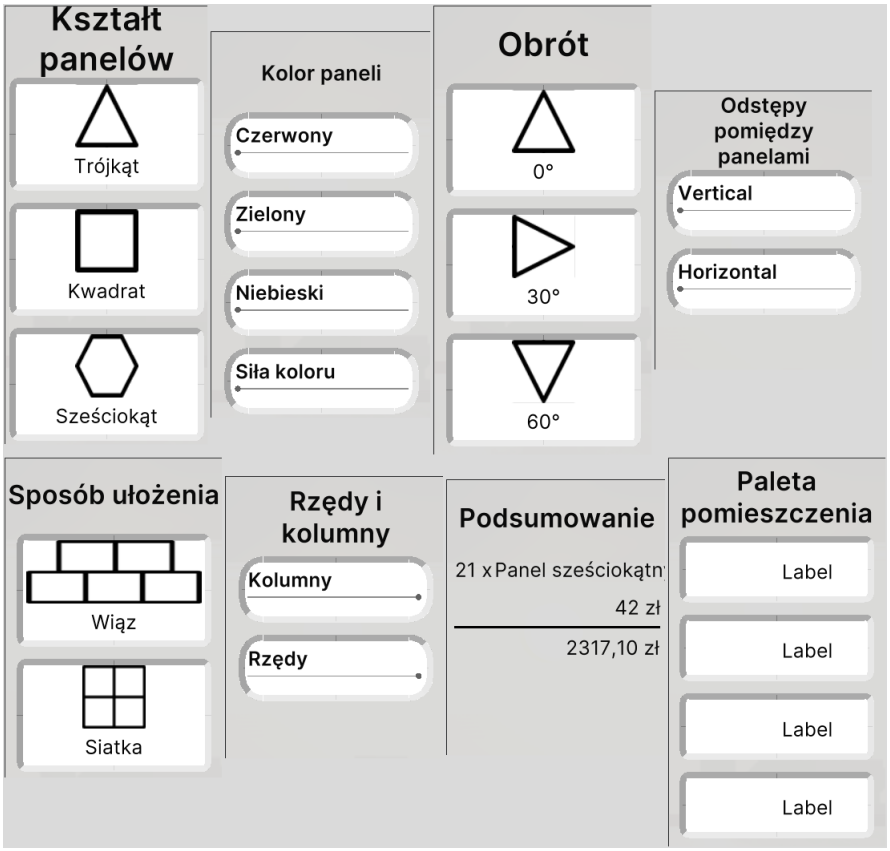
4.3.1 Interfejs użytkownika

Po wypisaniu wszystkich planowanych funkcjonalności i określeniu listy kolejnych kroków, przez które przechodzić będzie użytkownik, można zacząć projektować graficzny interfejs użytkownika. Układ i jego plan można rozrysować za pomocą różnych narzędzi. Można rozrysować jego plany na kartce papieru, za pomocą tekstu czy rysunku, ale można też użyć bardziej nowoczesnych narzędzi.

Do wstępnego rozrysowania kolejnych widoków w projekcie użyte zostanie internetowe narzędzie do tworzenia mock-upów, czyli uproszczonej graficznej wizji produktu końcowego. Będzie to jedynie atrapa, która nie będzie działać a ma

pomóc zaprojektować kolejne elementy. Podczas jego tworzenia jego styl jest uproszczony, a liczą się główny podział na kolejne, osobne widoki oraz elementy i ich planowane funkcjonalności.

Widok został podzielony na dwie części, wizualizacja z linią progresu, oraz widok z przyciskami i elementami interaktywnymi. Podczas procesu konfiguracji, kolejne widoki z przyciskami będą się zmieniać zależnie do obecnego stanu w procesie konfiguracji.



Rys. 7. Ułożenie elementów na poszczególnych widokach

Gdy zaprojektujemy już wstępnie kolejne widoki, można zacząć wydzielać z nich części wspólne, oraz elementy powtarzające się. Dzięki temu tworząc ten widok oszczędzić będzie można sobie niepotrzebnej pracy. Na przykład lewa część ekranu nie będzie się prawie w ogóle zmieniać, gdzie prawa będzie całkowicie zmieniana na inną podczas przejścia na inny etap konfiguracji. Każdy z widoków będzie zaprojektowany osobno i posiadać będzie inne funkcjonalności, każdy z nich będzie wyświetlany tym samym miejscu.

Do stworzenia graficznego interfejsu użytkownika użyty zostanie zestaw narzędzi UI Toolkit, pozwala on na łatwe i szybkie tworzenie tego elementu projektu. Jednym z narzędzi jest graficzny edytor za pomocą którego można składać interfejs z gotowych elementów. Każdy element następnie trzeba wyedytować, oraz dostroić za pomocą zbioru dostępnych do edycji właściwości.

Napierw podzielić można cały ekran na dwie części, zrobić to można poprzez ustawienie dwóch elementów obok siebie oraz odpowiednie ustawienie ich maksymalnych rozmiarów i proporcji wobec siebie nawzajem. Lewy z nich będzie miejscem na widok wizualizacji i pasek postępu, a prawy będzie posiadał jasne tło a jego zawartość będzie zmieniać się zależnie od etapu.

Wszystkie elementy są flexboxami, co oznacza że będą dynamicznie zmieniać swoje wymiary i ustawienie zależnie

od rozmiaru ekranu. Zdefiniować należy maksymalne i minimalne rozmiary elementów w pikselach albo procentach, a elementy samoczynnie zmieniać będa swoje rozmiary, z uwzględnieniem tych zasad. Dzięki temu aplikacja będzie mogą dostosowywać się do różnych rozmiarów i proporcji ekranów.

Na dole części z widokiem wizualizacji dodajemy osobny element w którym będzie pasek postępu konfiguracji. Będzie to prosty pasek na którego długości będą rozstawione kolejne etapy wraz z ikonami objaśniającymi ich funkcję. Po ukończeniu kolejnych etapów pasek będzie się stopniowo napełniał, a kliknięcie na przycisk przeniesie użytkownika do danego etapu konfiguracji. Pasek postępu jest stworzony z dwóch różnokolorowych prostokątów, które będą zmieniać swoje rozmiary, oraz kół z ikonami, które symbolizują poszczególne etapy.

Zawartość menu po prawej będzie się zmieniać zależnie od aktywnego etapu, przyciski przeznaczone dla każdego z nich będą przechowywane w osobnych kontenerach. Zależnie od sytuacji będzie stawał się widoczny lub niewidoczny. Każdy z nich posiada nagłówek wyjaśniający jaki aspekt produktu będzie aktualnie zmieniany. Pod spodem będzie miejsce na przyciski, suwaki oraz inne wymagane elementy interfejsu, potrzebne do dostrojenia tego jednego aspektu produktu.

Wybór palety barw będzie odbywać się poprzez naciśnięcie przycisku z odpowiadającą mu miniaturą i nazwą palety. Podobnie będzie z wyborem kształtu paneli, każdy przycisk posiadać będzie miniaturę z konturem danego kształtu. Zależnie od wybranego kształtu panelu, można będzie określić jak mają być obrócone panele. Nie wszystkie panele będą posiadały nawet taką opcję. W podobny sposób wybrać będzie można sposób ułożenia paneli, zależnie do wybranego kształtu dostępne będą sposoby ich ułożenia.

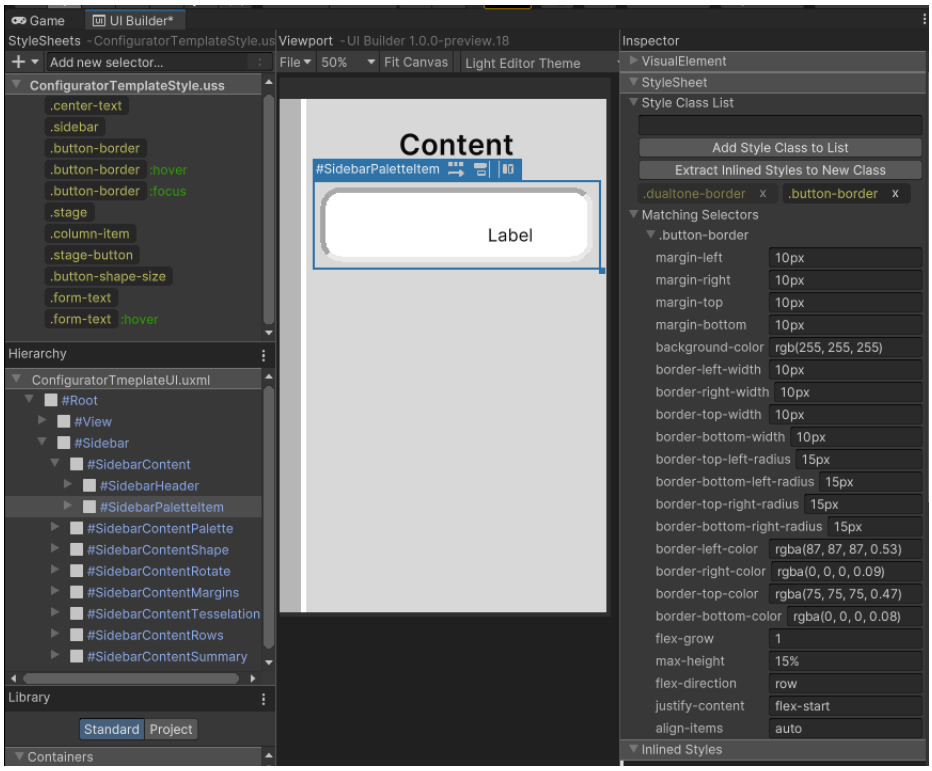
Za pomocą suwaków będzie można zdefiniować odstępy pomiędzy panelami, oraz liczbę rzędów oraz kolumn paneli. Odstępy będą mogły być dowolną liczbą z podanego zakresu, a liczba paneli będzie liczbą całkowitą.

Przedostatnim widokiem będzie ekran podsumowania, który wylistuje nam dokładną nazwę wybranego panelu, ilość, kolor, jego cenę za sztukę oraz podliczoną cenę za wszystkie panele na danej ścianie.

Ostatnim widokiem będzie widok formularza kontaktowego, w którym wprowadzić będzie można swoje dane kontaktowe, dzięki którym będzie można się skontaktować z klientem. Polami tekstowymi do wypełnienia będą imię, nazwisko, adres email, oraz numer telefonu do klienta. Po wypełnieniu formularza, wysłać go będzie można klikając przycisk 'Wyślij'.

Ważnym elementem ułatwiającym stylowanie elementów interfejsu są arkusze stylów oraz klasy które one definiują. Pliki Unity Style Sheet (USS), które pełnią podobną rolę co pliki CSS w stronach internetowych, mogą zostać użyte do wypisania zasad, które definiować będą wygląd elementów. Raz zdefiniowana klasa, może zostać przypisana dowolnemu elementowi, dzięki czemu zyskuje on cechy wyglądu przypisane danej klasie. Jest to ogromne ułatwienie, jednak pliki USS nie są pozbawione wad. Pamiętać należy, że mechanizm ten nie jest tak samo dogłębnie rozbudowany co wersja dla stron internetowych i pewne parametry elementów znane z CSSa nie są zaimplementowane w USS. Przykładem są cienie pod elementami, które można zdefiniować w plikach CSS, ale Unity już tekiej opcji nie wspiera.

Przykładem w projekcie jest stylowanie przycisku, poprzez zdefiniowanie klasy .button-border, która tworzy różnokolorowe obramowanie wokół przycisku. Klasa ta jest dodawana do każdego elementu, który działa jako przycisk dzięki czemu jego wygląd kojarzy się użytkownikowi z określoną akcją naciśnięcia na niego. Po prawej stronie ekranu widoczna jest lista właściwości które są zmieniane w danej klasie.



Rys. 8. Podgląd stylowania przycisku w edytorze interfejsu użytkownika

Tworzenie ikon i miniatur

Na potrzeby interfejsu użytkownika wymagane jest stworzenie miniatur przedstawiających geometryczne kształty paneli, obrócone ich wariacje oraz alternatywne sposoby ich ułożenia. Miniatury te będą grafiką wektorową wykonaną w programie Inkscape. Stworzenie ich jednocześnie jest wymagane, aby obrazy te posiadały te same rozmiary, grubość linii, oraz styl graficzny.

Narysowanie podstawowych figur geometrycznych w programie do grafiki wektorowej, Inkscape jest trywialne proste, ze względu na wbudowane narzędzie do tworzenia figur równobocznych o zadanej ilości boków. Po narysowaniu kształtu możliwe jest ustawienie koloru i grubości linii, która go tworzy. Na końcu eksportujemy wszystkie figury jako osobne pliki graficzne.

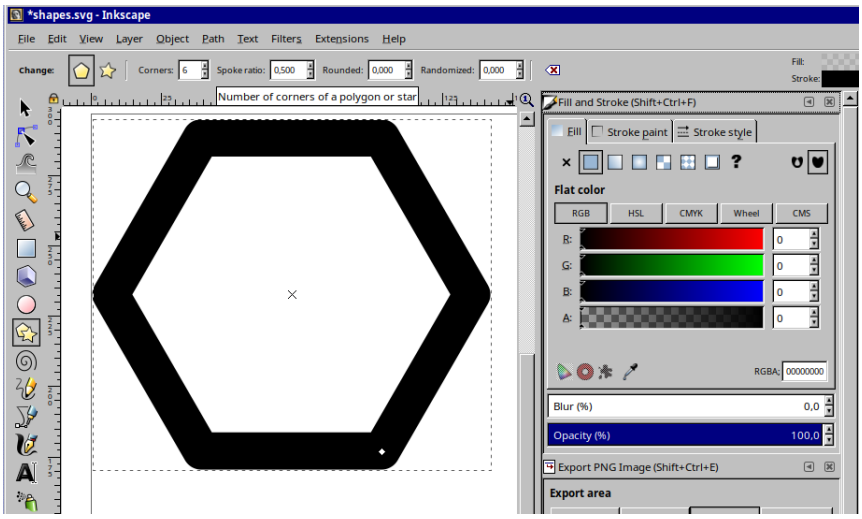
4.3.2 Skrypty implementujące logikę programu

Logika programu jest podzielona na parę skryptów, gdzie każdy z nich realizować będzie inną funkcjonalność.

Pisanie skryptu ściany interaktynwej

Najważniejszym skryptem w projekcie jest skrypt ustawiający panele na ścianie interaktywnej. Ustawione będą mogły być panele o trzech kształtach: trójkątnym, kwadratowym i sześciokątnym, każdy z nich będzie ustawiany w trochę inny sposób, który wymaga trochę innego podejścia. Panele będą mogły być obrócone, a niektóre z nich będą mogły zostać ustawione w siatce, albo w wiąz.

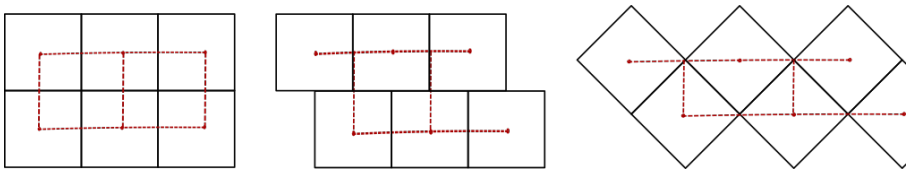
Ściana interaktywna będzie obiektem posiadającym płaską powierzchnię o skończonych wymiarach. Skrypt ten będzie dołączany do niej, oraz implementować będzie funkcjonalność nakładania na nią paneli wedle zadanych parametrów.



Rys. 9. Tworzenie wielokąta w programie Inkscape

Sama powierzchnia będzie niewidoczna, a cały obiekt ściany interaktywnej będzie ustawiany na ścianach pomieszczenia.

Na początku wymagane będzie określenie obliczenie maksymalnej liczby kolumn i rzędów, które zmieszczą się na danej ścianie. Wystarczy podzielić wymiary ściany przez wymiary panelu dodając do niego opcjonalne odstępy pomiędzy nimi. Za pomocą wbudowanych funkcji silnika Unity3d można odczytać rzeczywiste wymiary ściany interaktywnej, oraz panelu który będzie ją pokrywał. Ważne jest wtedy branie pod uwagę sposobu w jaki obrócony ma być panel, bo jego obrót nie jest brany pod uwagę podczas liczenia wymiarów modelu trójwymiarowego. Zależnie od wybranego kształtu panelu jego wymiary po obróceniu będą inne, dlatego już na tym etapie trzeba zrobić osobne przypadki dla różnych kształtów oraz różnych ich wariantów obrotu.



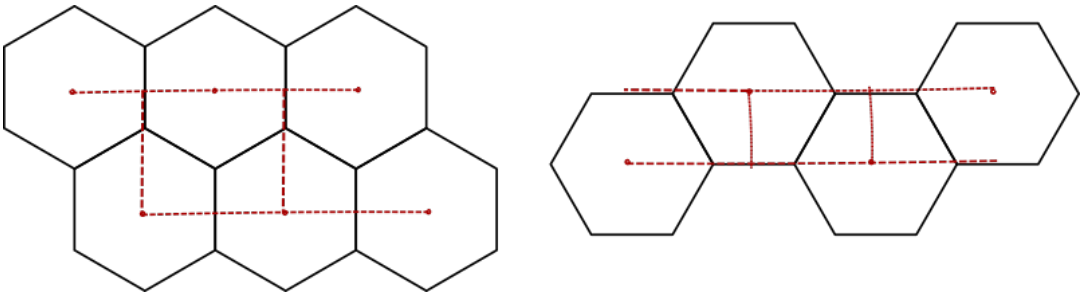
Rys. 10. Różne sposoby ułożenia parkietażu kwadratu

Po obliczeniu wymiarów, można określić wymiary siatki wedle której ustawiane będą panele, bo nie zawsze siatka jest tych samych wymiarów co wymiary panelu. Kwadraty są prostym przykładem, który sie komplikuje kiedy obróci się je o 45 stopni, wtedy wysokość siatki zmniejsza się o połowę, gdyż panele zaczynają się przeplatać ze sobą.

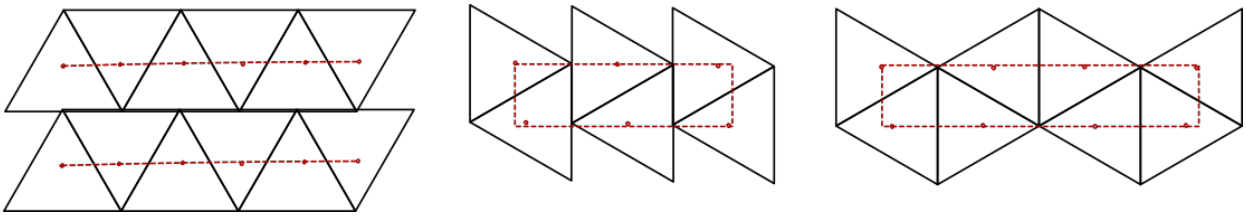
Dodatkową zmianą jest fakt, że panele nie zawsze są poustawiane jeden pod drugim. W przypadku zerowej rotacji kwadratu jest to styl ułożenia w wiąz, a w przypadku panelu obróconego o 45 stopni jest to jedyny sposób na ułożenie ich.

Podobnym przypadkiem są sześciokąty, które zawsze muszą być ustawione naprzemiennie, dodatkowo wymiary siatki różnią się od kształtu panelu, tak jest również w przypadku sześciokąta. Normalnie szerokość siatki jest równa szerokości panelu, a jej wysokość trzecich ósmych jego wysokości ze względu na przeplatanie się ze sobą paneli z kolejnych rzędów. Po obróceniu sześciokąta, szerokość siatki robi się większa od szerokości panelu, po obróceniu.

Najbardziej skomplikowane jest ułożenie paneli o kształcie trójkąta równobocznego. Każdy następny panel musi być obrócony o 180 stopni wobec poprzedniego, a szerokość siatki jest połową szerokości panelu. W przypadku obróconego



Rys. 11. Różne sposoby ułożenia parkietażu sześciokąta



Rys. 12. Różne sposoby ułożenia parkietażu trójkąta

panelu, obrót o 180 stopni tyczy się co drugiego rzędu.

Panele są ustawiane od lewej do prawej, rzędami od góry do dołu. Najpierw wybierana jest początkowa pozycja ściany interaktywnej, która jest jej środkiem. Potem na podstawie wymiarów panelu, liczby rzędów i kolumn i odstępów pomiędzy nimi, obliczany jest punkt początkowy. Następnie w zagnieżdżonej pętli ustawiane są panele, wedle rzędów i kolumn. Tworzenie panelu odbywa się poprzez budowaną w silnik Unity3d funkcję 'Instantiate', która tworzy nową instancję obiektu z podanego prefabrykatu w podanej przez nas pozycji i obrocie. Dla zachowania porządku wszystkie stworzone panele są dziećmi pustego elementu o nazwie 'panels', ułatwi to potem usunięcie ich wszystkich jednocześnie i stworzenie ich na nowo.

Kiedy chcemy zmienić sposób ułożenia paneli, najpierw parametry ustawienia paneli są zmieniane, uruchamiana jest funkcja, która usuwa obiekt 'panels', wraz ze wszystkimi panelami. Następnie należy obliczyć na nowo wartości wymaganych zmiennych, a dopiero potem można ponownie uruchomić funkcję ustawiającą panele na swoich miejscach.

Stworzone zostały funkcje pozwalające na zmianę: kształtu panelu, rotacji panelu, sposobu ułożenia, szerokości i wysokości odstępów i liczby kolumn i rzędów. Każda z nich oprócz nadpisywania parametrów, usuwa panele na danej ścianie i buduje je na nowo z uwzględnieniem nowej wartości parametru. Funkcje te będą bezpośrednio uruchamiane przez interfejs użytkownika.

Pisanie skryptu dla kamery, oraz punktów obserwacji

Ustawienie kamery w wirtualnej scenie będzie kluczowe dla odpowiedniego pokazania wizualizacji, dlatego jej ustawienie jest tak ważne. Kamera będzie zawsze skierowana w kierunku aktualnie edytowanej ściany, możliwe będzie przybliżanie i oddalanie, oraz przemieszczanie jej w ustalonym zakresie. Przybliżanie i oddalanie widoku odbywa się poprzez zmniejszanie i zwiększanie odległości od ściany, a poruszanie kamerą, odbywa się poprzez obrót kamery, względem ściany, gdzie punktem przyłożenia tego obrotu jest środek obserwowanej ściany. Poprzez przyciśnięcie przycisku myszy i poruszenie kursora poruszana będzie kamera, która cały czas zwrócona będzie w kierunku ściany. Oddalanie i przybliżanie kamery będzie przypisane do kółka myszki.

```
private void BuildPanels(){
    panels = new GameObject("panels");
    panels.transform.rotation = gameObject.transform.rotation;
    panels.transform.parent = gameObject.transform;

    Vector3 startingPoint = gameObject.transform.position;
    startingPoint += gameObject.transform.forward*(columns-1)*gridX/2;
    startingPoint += gameObject.transform.right*(rows-1)*gridY/2;
    if(staggered){
        startingPoint+= gameObject.transform.forward*gridX/4;
    }
    if(panel.name=="Panel.Triangle" && panelRotation==30){
        startingPoint+= gameObject.transform.forward*gridX/4;
    }

    Vector3 panelPosition = startingPoint;
    for (int j = 0; j < rows; j++){
        panelPosition = startingPoint;
        panelPosition -= gameObject.transform.right * j * gridY;
        if((staggered || panel.name=="Panel.Triangle" && panelRotation==30) && j%2==0){
            panelPosition-= gameObject.transform.forward*gridX/2;
            if(panel.name=="Panel.Triangle"){
                panelPosition-= gameObject.transform.forward*gridX/2;
            }
        }

        for (int i = 0; i < columns; i++){
            Vector3 tempPosition = panelPosition;
            if(panel.name=="Panel.Triangle" && panelRotation==30 && !staggered){
                int mod=1;
                if(i%2==1)mod=-1;
                Debug.Log("Triangle MAGIC!!!");
                tempPosition -= mod*gameObject.transform.forward*horizontalOffset;
                tempPosition += mod*gameObject.transform.right*(verticalOffset/2);

            }
            if(panel.name=="Panel.Triangle" && panelRotation==30 && staggered){
                int mod=1;
                if(j%2==1)mod=-1;
                Debug.Log("Triangle MAGIC!!!");
                tempPosition += mod*gameObject.transform.forward*horizontalOffset;
                tempPosition += mod*gameObject.transform.right*(verticalOffset/2);

            }

            GameObject tempPanel = Instantiate(this.panel, tempPosition, gameObject.transform.rotation);
            tempPanel.transform.Rotate(0,panelRotation,0,Space.Self);
            if(panel.name=="Panel.Triangle" && i%2>0 && (!staggered || panelRotation==0)){
                tempPanel.transform.Rotate(0,cyclingRotation,0,Space.Self);
            }
            if(panel.name=="Panel.Triangle" && j%2>0 && staggered){
                tempPanel.transform.Rotate(0,cyclingRotation,0,Space.Self);
            }
            tempPanel.transform.parent = panels.transform;
            panelPosition-= gameObject.transform.forward*gridX;
        }

    }
}
```

Rys. 13. Fragment kodu odpowiedzialny za ustawianie paneli na ścianie

```
// Update is called once per frame
0 references
void Update()
{
    if (Input.GetMouseButtonDown(1)){
        Cursor.lockState = CursorLockMode.Locked;
    }else if(Input.GetMouseButtonUp(1)){
        Cursor.lockState = CursorLockMode.None;
    }

    if(Cursor.lockState == CursorLockMode.Locked){
        float mouseX = Input.GetAxis("Mouse X");
        float mouseY = -Input.GetAxis("Mouse Y");

        rotationX = mouseX * cameraSensitivity * Time.deltaTime;
        rotationY = mouseY * cameraSensitivity * Time.deltaTime;

        transform.transform.Rotate(Vector3.left*rotationX, Space.Self);
        transform.transform.Rotate(Vector3.forward*rotationY, Space.Self);

        transform.transform.localEulerAngles = ClampVector(transform.transform.localEulerAngles);

        GetComponentInChildren<Camera>().transform.LookAt(
            currentWaypoint.interactiveWall.gameObject.transform.position,Vector3.up
        );
    }

    if(Input.mouseScrollDelta.y !=0 ){
        Debug.Log("Scroollininio: "+Input.mouseScrollDelta);
        setCameraDistance(Input.mouseScrollDelta.y);
    }

    if( Input.GetMouseButtonDown(0) ){
        Ray ray = gameObject.GetComponentInChildren<Camera>().ScreenPointToRay( Input.mousePosition );
        RaycastHit hit;

        if( Physics.Raycast( ray, out hit, 100 ) )
        {
            if( hit.transform.gameObject.GetComponentInChildren<Wall>() ){
                setCurrentWaypoint( hit.transform.GetChild(0).gameObject.GetComponentInChildren<CameraWaypointScript>() );
                Initialize();
            }

        }
    }
}
1 reference
```

Rys. 14. Fragment kodu obsługującego ruch kamery myszką

Sposób poruszania kamery będzie definiowany i implementowany przez skrypt do niej dołączony. Co każdą klatkę sprawdzany będzie stan wciśnięcia przycisku myszy, przesunięcie kursora, oraz obrót kółka myszy. Jeśli wykryte będą jakieś zmiany uruchamiana będzie odpowiednia akcja: przesunięcia kamery, oddalenia lub przybliżenia jej, albo zmiany obecnie aktywnej ściany interaktywnej.

Możliwa jest zmiana obecnie edytowanej ściany poprzez wskazanie jej kursorem i kliknięcie na nią. Punkt przyłożenia kamery, wraz z kamerą i jej pozycją zostaną przypisane do nowej interaktywnej ściany. Odległość kamery od ściany jest ustawiana na minimalną przewidywaną odległość, a kamera wycentrowana.

Kamera będzie poruszała się w określonym zakresie kąta względem ściany, osobno dla odchylenia w lewo, prawo, w górę i dół, informacje te będą zmieniały się zależnie od obecnie aktywnego skrpytu punktu obserwacji.

Skrypt punktu obserwacji będzie skryptem dołączanym do każdej ściany interaktynwej, będzie on jedynie przetrzymywał zmienne określające maksymalne kąty wychylenia kamery dla danej ściany interaktywnej i przewidywany zakres odległości kamery do ściany. Wartości będą ręcznie wpisane dla każdej ściany z kolei.

Pisanie skryptu zarządzającego globalnym oświetleniem

Implementacja oświetlenia będzie realizowana poprzez dwa skrypty tego dotyczące. LampScript będzie skryptem dołączanym do każdego obiektu źródła światła pozwalającym na zarządzanie nim, a kontroler oświetlenia będzie zarządzać wszystkimi elementami posiadający poprzednio opisany skrypt.

Oświetlenie w całej scenie będzie zarządzane przez jeden skrypt, kontroler oświetlenia. Skrypt ten będzie zarządzał pojedynczym źródłem światła imitującym słońce, oraz grupą mniejszych źródeł światła umieszczonych w lampach, odpowiedzialnych za nocne oświetlenie. Wszystkie lampy będą zgrupowane w jednym obiekcie nazwanym 'lamps'. Dodatkowo dodany został trzeci obiekt, odpowiedzialny za oświetlenie dodatkowe. Są to jedyne zmienne publiczne zadeklarowane w tym skrypcie.

```
public class LightsController : MonoBehaviour
{
    [SerializeField]
    2 references
    public GameObject Sun;
    [SerializeField]
    0 references
    public GameObject AmbientLight;
    [SerializeField]
    2 references
    public GameObject Lamps;


    //Function sets all lamps imitating tiem of day
    2 references
    public void setDay(){
        LampScript[] lamps = Lamps.GetComponentsInChildren<LampScript>();
        foreach (LampScript lamp in lamps){
            lamp.setBrightness(0);
        }
        Sun.GetComponent<LampScript>().setBrightness(1);
    }

    1 reference
    public void setNight(){
        LampScript[] lamps = Lamps.GetComponentsInChildren<LampScript>();
        foreach (LampScript lamp in lamps){
            lamp.setBrightness(1);
        }
        Sun.GetComponent<LampScript>().setBrightness(0);
    }
    0 references
    void Start()
    {
        setDay();
    }
}
```

Rys. 15. Kod implementujący kontroler oświetlenia

Skrypt posiada jedynie dwie funkcje: setNight(), setDay(). Są one odpowiedzialne za włączanie i wyłączanie określonych źródeł światła, aby stworzyć iluzję światła dziennego rzucającego cienie przez okna, albo imitować światło emitowane przez lampy we wnętrzu pomieszczenia. Kontroler oświetlenia realizuje włączanie i wyłączenie poszczególnych świateł poprzez skrypt lampScript dołączany do obiektów sceny posiadających źródło światła.

LampScript jest skryptem dołączanym do wszystkich elementów sceny posiadającym źródło światła. Zaimplementowane funkce: setColor(), setBrightness() pozwalają na zmianę koloru emitowanego światła, oraz procentową zmianę jego natężenia. Siła świecenia poszczególnych lamp może być różna a regulowane natężenie proporcjonalnie zmniejsza ich siłę względem początkowej wartości.

Pisanie skryptu zarządzającego paletą pomieszczenia

Zmiana palety całego pomieszczenia odbywa sie poprzez dwa skrypty. Kontroler palety pomieszczenia, oraz skrypt przetrzymujący materiały. Konteroler palety posiada jedną funckję setGlobalPalette(), która wyszukuje wszystkie instancje skryptu pomocniczego, PaletteScript oraz uruchamia w nich funkcję zmiany palety. Funkcja ta jest uruchamiana przez interfejs użytkownika podczas wybierania określonej palety wnętrza.

Istnieje tylko jeden kontroler palety i jest on dołączony do głównego obiektu sceny. Skrypty palety kolorów, są dołączone do każdego obiektu którego materiały mają się zmieniać.

Skrypt PaletteScript jest dołączany do każdego obiektu sceny, którego materiały mają się zmieniać po zmianie palety pomieszczenia. Posiada on zadeklarowane tabele, w których przetrzymywane są kolejne materiały, które odpowiadają wyglądowi obiektu w różnych paletach wnętrz. Funkcje w nim zaimplemntowane: setMaterial(), setColor() pozwalają na zmianę materiałów danego obiektu i pokolorowanie obecnie wybranego materiału.

Pisanie skryptu zarządzającego interfejsem użytkownika

Na interfejs użytkownika składa się parę plików, plik o rozszerzeniu UXML, w którym zadeklarowana jest struktura i ustawienie elementów interfejsu, plik USS, który definiuje zasady dotyczące wyglądu tych elementów, oraz skrypt, który wchodzić będzie w interakcję z resztą projektu

W skrypcie obsługującym interfejs użytkownika, zadeklarowane są zmienne odpowiadające elementom interfejsu, zadeklarowane również będą funkcje uruchamiane po naciśnięciu elementów interfejsu. Wymagane jest również przypisanie przycisków interfejsu do odpowiadających im funkcji, aby przypisać interakcję elementu interfejsu do przypisanej jemu akcji.

Na początku pliku zadeklarowane zostały zmienne odpowiadające przyciskom odpowiadającym poszczególnym etapom konfiguracji produktu. Po uruchomieniu programu, elementy interfejsu użytkownika zostaną przypisane do tych zmiennych, następnie do tych zmiennych przypisane zostają funkcje odpowiedzialne za uruchamianie kolejnych etapów konfiguracji.

Funkcje zmieniające etap konfiguracji mają proste zadanie, ukryć wszystkie elementy interfejsu, które nie są potrzebne na tym etapie i pokazać te obecnie potrzebne, oraz w razie potrzeby wykonać jakieś dodatkowe akcje, takie jak aktualizowanie parametrów interfejsu. Elementy interfejsu odpowiadające poszczególnym etapom, zgrupowane są w osobnych obiektach, dzięki czemu łatwo jest je wszystkie ukryć, albo odkryć w razie potrzeby.

Wymagane jest też zdefiniowanie i przypisanie akcji do elementów mieszczących się w zgrupowanych kategoriach, odpowiadającym poszczególnym etapom. Stworzone zostały funkcje uruchamiające dane akcje, a funkcje te przypisane zostały do odpowiadających im przycisków.

Pierwszym etapem jest wybranie palety pomieszczenia, dlatego znajdujemy obiekt grupujący pod sobą wszystkie przyciski zmieniające paletę pomieszczenia oraz do każdego z kolei przypisujemy akcję, uruchamiania funkcji zmieniającej palety pomieszczenia. Dla każdego z przycisków uruchamiana jest ta sama funkcja, lecz z inną wartością parametru, dzięki temu, możemy wszystko wykonać za pomoca pętli iterując po kolei przez wszystkie przyciski. Uruchamiana funkcja, setGlobalPalette(), która została zdefiniowana w kontrolerze palety pomieszczenia, oraz jako argument przyjmuje indeks aplikowanej palety materiałów.

Kolejnym etapem jest wybranie kształtu paneli. Każdemu kształtowi przypisany jest osobny przycisk opatrzony

```
_formButton = _ui.rootVisualElement.Q<Button>("StageFormButton");
_formButton.clicked += FormButtonOnClicked;

//adding palette handlers for many buttons
VisualElement sidebarPalette = _ui.rootVisualElement.Q<VisualElement>("SidebarContentPalette");
int counter=0;
foreach(var item in sidebarPalette.Children() ){
    Debug.Log(item.name+" type: "+item.GetType().ToString()+" counter:"+counter);
    if(item.GetType().ToString() == "UnityEngine.UIElements.Button"){
        Button tempButton = (Button)item;
        int temp = counter;

        item.RegisterCallback<ClickEvent>(
            ev => GameObject.FindObjectOfType<PaletteControllerScript>().setGlobalPalette(temp)
        );
        counter++;
    }
}
```

Rys. 16. Kod przedstawiający różne sposoby na przypisanie akcji do przycisku

obrazkiem i nazwą figury geometrycznej. Za pomocą pomocniczej funkcja pobieramy skrypt aktualnie edytowanej ściany interaktywnej. W niej zadeklarowana jest funkcja, która jako argument przyjmuję liczbę porządkową przypisaną każdemu z kształtów. Usuwa ona wszystkie panele, zmienia ona kształt panelu na ten przez nas wybrany i na nowo oblicza wszystkie zmienne oraz buduje wszystkie panele od nowa, na wybranej ścianie.

W następnym etapie wybrać można z jakiego materiału wykonane mają być panele dekoracyjne. Materiał wybiera się poprzez naciśnięcie przycisku. Każdy przycisk posiada nazwę materiału, któremu odpowiada. Każdy z nich uruchamia tę samą funkcję mieszczącą się w skrypcie PanelColor, lecz z inną wartością parametru index, będący liczbą porządkową danego materiału. Skrypt PanelColor dołączany jest do każdej ściany interaktywnej, oraz pozwala na zmianę koloru i materiału z którego wykonane są panele.

Kolejnym parametrem do określenia jest kolor paneli ściennych, odbywa się to za pomocą czterech suwaków. Pierwsze trzy suwaki po kolei odpowiadają za ilość: czerwonego, zielonego i niebieskiego pigmentu w kolorystyce paneli. Ostatnim osobnym suwakiem ustawić można stopień nasycenia wybranego koloru. Każdy z suwaków uruchamia tę samą funkcję lokalną setColor(), lecz z inną wartością parametru, którego wartość odpowiada osobnym kolorom. Funkcja ta najpierw pobiera obecny kolor, oraz nadpisuje wartość jednego z kolorów, wartością pobraną z odpowiedniego suwaka. Zmiana koloru materiału odbywa się przez funkcję znajdująca się w skrypcie ColorScript.

Następnie można wybrać kąt obrotu poszczególnych paneli, dla każdego kształtu wybór opcji jest inny, a przyciski posiadają obrazki pokazujące daną figurę geometryczną po obrocie o zadaną ilośc stopnii. Dodatkowo tekst objaśnia odkładną wartość obrotu paneli. Po przejściu do etapu wyboru obrotu panelu, sprawdzany jest obecnie wybrany kształt, na jego podstawie pokazywane są przyciski pokazujące jedynie te opcje, które odpowiadają wybranemu kształtowi. Każdy z przycisków uruchamia tę samą funckję, setPanelRotation() zadeklarowaną w skrypcie ściany interaktywnej, która jako argument przyjmuje nową wartość stopni obrotu paneli. Funkcja ta usuwa stare panele, nadpisuje wartość ich rotacji oraz na nowo ustawia na ścianie obrócone już panele.

W osobnym etapie można ustawić wartości pionowych i poziomych odstępów pomiędzy panelami. Odbywa się to poprzez dwa osobne suwaki. Przesunięcie ich uruchamia funkcję pomocniczą, która odczytuje ich wartość, oraz uruchamia funkcję setHorizontalPanelGaps(), albo setVerticalPanelGaps() zadeklarowane w skrypcie ściany interaktywnej.

Następnie można wybrać alternatywny sposób ułożenia paneli, który zależny jest od kształtu panelu oraz jego rotacji. Pozwala on ustawić panele w siatkę, albo w wiąz. Alternatywna opcja jest dostępna dla kwadratów z zerowa rotacją i trójkątów. Przyciski posiadają obrazki które przedstawiają sposób ułożenia paneli.

Osobna grupa elementów pozwala na ustalenie liczby kolumn i rzędów paneli, która odbywa się za pomoca suwaków. Podczas budowania paneli na ścianie, zostaje obliczona maksymalna liczba rzędów i kolumn paneli, które się zmieszczą na ścianie. Maksymalna wartość jaką można wybrać za pomocą suwaka jest maksymalną liczbą która się zmieści na ścianie, jest ona ustanawiana po kliknięciu przycisku uruchamiający ten etap. Po każdej akcji kliknięcia na suwak, uruchamiania jest funkcja która odczytuje jego nową wartość, zmienia parametry ściany interaktywnej, czyści stare panele i buduje je na nowo.

Etap podsumowywujący przedstawia liczbę, nazwę, oraz cenę panelu. Pokazywana jest również całkowita cena zamówienia. Każda z tych informacji, pokazywana jest za pomocą osobnego elementu z tekstem. Podczas uruchomienia tego etapu, zależnie od kształtu ustawiana jest odpowiednia nazwa panelu, ilość paneli jest liczona na podstawie ilości kolumn i rzędów, a sumaryczna cena zamówienia jest obliczana z pomnożenia ceny panelu i ich liczby. Liczba paneli i aktualna jego nazwa jest pobierana ze skryptu aktualnie aktywnej ściany interaktywnej.

Ostatni etap jest formularzem do wypełnienia dla klienta, posiada on pola tekstowe do których klient wpisuje swoje dane kontaktowe. Dla każdej informacji: imienia, nazwiska, numeru telefonu i adresu email przygotowane jest osobne pole. Przycisk 'Wyślij formularz' finalizuje cały proces.

4.3.3 Tworzenie wirtualnej sceny i finalizacja projektu z użyciem edytora Unity

Pierwszą większą rzeczą wykonaną w edytorze Unity3D będzie stworzenie sceny. Podzielona ona zostanie na grupy obiektów dla zachowania porządku struktury projektu. Wszystkie elementy składające się na pomieszczenie będą przypisane do jednego obiektu, dzięki temu wykonując na nim akcje przemieszczania, obracania i skalowania będą one również zmieniały wszystkie obiekty jemu podległe, czyli wszystkie elementy pomieszczenia. Coś podobnego zastosowane będzie do zgrupowania ze sobą obiektów o podobnym przeznaczeniu, przykładem mogą być ściany, albo meble.

Najpierw tworzymy nową płaską powierzchnię na którą nakładamy teksturę z diagramem przedstawiającym układ wnętrza pokoju oraz rozmieszczenie mebli. Potem skalujemy powierzchnię, aby miała te same proporcje co zdjęcie z teksturą, dzięki temu tekstura zachowa swoje pierwotne proporcje. Powierzchnię z planem używać będziemy jako zdjęcia referencyjnego, na podstawie którego ustwawimy ściany pomieszczenia, meble oraz kolejne punkty dla kamer.

Następnie importujemy modele do projektu oraz rozpakowywujemy je, następnie z każdego mebla tworzymy prefabrykat, z których będziemy tworzyć pomieszczenie. Zaczynamy od tworzenia ścian pomieszczenia, puste ściany możemy skalować do dowolnej wymaganej szerokości, dzięki temu nie tworzymy niepotrzebnych dodatkowych elementów. Elementy ścian posiadające otwory na okna, nie mogą być skalowane tak łatwo, ale pomieszczenie używa okien o standardowych rozmiarach, więc i one takie pozostaną.

Gdy ściany pomieszczenia są już na swoim miejscu można rozstawić meble na swoje miejsca. Bardzo pomocny staje się plan pomieszczenie, na którym meble również zostały narysowane, dzięki czemu można rozstawić meble na swoje miejsca, oraz w razie potrzeby nawet w pewnym stopniu przeskalować meble aby bardziej pokrywały się z planem.

Etapem finalizującym cały projekt jest podłączenie napisanych skryptów do interfejsu użytkownika, to on będzie uruchamiał każda akcję dziejącą się w świecie gry.

5 Podsumowanie

Stworzony program spełnia postawione przed nim wymagania w zakresie zautomatyzowania procesu zapoznania klienta z ofertą i odciążenie w ten sposób pracowników obsługi klienta, przechodząc przez kolejne etapy konfiguracji bardziej określa on swoją wizję. Kolejnym aspektem jest pokazanie klientowi produktu w odpowiednim środowisku, które pozwala na lepsze wizualnie zapoznanie się z produktem i przyrównanie go rozmiarami i kolorystyką do wnętrza, dzięki zmiennej kolorystyce wnętrza i kolorom paneli jest to możliwe.

Użycie platformy posiadającej narzędzia do tworzenia gier rozwiązało problem implementacji większości funkcjonalności, przy czym wymagana była wiedza na temat użycia narzędzi Unity. Dzięki temu możliwe było stworzenie sceny, materiałów i interfejsu użytkownika, bez wymaganej wiedzy na temat ich tworzenia, dzięki ich przejrzystemu interfejsowi graficznemu. Jego intuicyjność pozwoliła na wygodne i szybkie wykonanie zadania. Ułatwieniem były wbudowane funkcje i biblioteki, które implementują dokładnie te funkcje, które potrzebne były w projekcie, dzięki czemu nie było wymagane implementowanie funkcjonalności od początku do końca, wystarczyło tylko dopisać brakujące fragmenty kodu tworzącę logikę programu. Możnaby to zadanie ułatwić jeszcze bardziej używając gotowych już modeli trójwymiarowych, zamiast tworzyć każdy jeden z nich od podstaw, oszczędziłoby to sporo czasu, ale mimo tego warto było się zapoznać z tworzeniem modeli od początku do końca.

Podczas fazy planowania projeku wszystko wydaje się być o wiele prostrze, niż jest w rzeczywiskości. Nie jest możliwe przewidzenie wszyskich problemów i zawiłości, które samoczynnie wyjdą podczas implementacji projektu. Wnioski tutaj wypisane są zapisem rzeczy, które nie zostały przewidziane podczas fazy projektowania programu, a były potrzebne w fazie jego implementacji.

Wydawać by się mogło, że podczas pisania programu komputerowego najwięcej czasu zabiera samo pisanie kodu, lecz wbrew pozorom najwięcej czasu zabrało rozplanowanie i stworzenie materiałów, niż samo jego pisanie. Prawdopodobnie wynika to z faktu, że nie było wymagane pisanie żadnych funkcjonalności od podstaw, a używanie tych, które zapewnia nam silnik do tworzenia gier komputerowych. Dodatkowym czynnikiem przyspieszającym tworzenie kodu, był fakt, że już wcześniej byłem zapoznany z podstawami tego silnika, jedynie tworzenie UI przysporzyło problemów z uwagi na brak wcześniejszego doświadczenia tym zestawem narzędzi.

Podczas projektowania interfejsu użytkownika została użyta webowa aplikacja UIzard, stworzona makieta miała pomóc określić ogólny podział interfejsu na części, na kolejne ekrany i ustalić styl tworzonego interfejsu. Inspiracją miał być, obecnie trendy styl neumorficzny, który wyróżnia się użyciem jednakowych kolorów, gdzie poszczególne elementy są zaznaczone za pomocą trójwymiarowych cieni. Wygląda to trochę jakby w jednakowej płycie metalu zostałyby wybite elementy interaktywne. UIzard pozwalał na dodanie jednolitego, rozmazanego cienia do każdego elementu, niestety użyta w projekcie biblioteka do tworzenia interfejsu użytkownika UI Toolkit, nie implementuje takiej funkcjonalności. W parametrach elementów interfejsu nie zostały zaimplementowane niektóre właściwości języka CSS uważane za podstawowe. Wymagane było zmiennie troszkę stylu, przez co tworzenie interfejsu użytkownika zajęło więcej niż przewidywano.

Wersja UI Toolkit użyta w projekcie posiada dopisek 'preview' co oznacza że jest to mocno wczesna wersja, przeznaczona do zapoznania się użytkowników z danym programem, a dla programistów jest okazją do zebrania opinii innych na temat tego oprogramowania. Narzędzie czasami zawieszało cały system operacyjny (linux), bez żadnego ostrzeżenia, co skutkowało utratą niezapisanych zmian w edytorze interfejsu.

Kolejnym nieoczekiwanym problemem okazało się obracanie paneli podczas ich ustawiania. Planowanym algorytmem ich ustawiania było ich ustawienie w przestrzeni, a potem obrócenie. Błędne jednak było założenie, że oś obrotu figury będzie automatycznie wyznaczana prawidłowo. Domyślnie punkt obrotu modelu jest automatycznie wyznaczany przez silnik. W przypadku panelu w kształcie kwadratu i sześciokąta, odbywało się to prawidłowo. Punkt ich obrotu pokrywał się z ich środkiem ciężkości. Niestety punkt ten był błędnie wyznaczany w przypadku panelu trójkąta równobocznego. Rozwiązaniem było stworzenie pustego elementu nadrzędnego, oraz dodanie do niego panelu w kształcie trójkąta z odpowiednim przesunięciem względem obiektu nadrzędnego. Dzięki temu obracając obiekt nadrzędny, obracamy również panel wokół nowego punktu obrotu, który pokrywa się z środkiem ciężkości panelu.

Rozplanowanie całego projektu bardzo ułatwia i przyspiesza pracę, przydatna bardzo była webowa aplikacja Miro, w której można tworzyć różnego rodzaju listy, tabele, diagramy i inne bardziej zawansowane struktury. Jedną z nich była tablica kanban, która pozwala na lepsze usystematyzowanie pracy and projektem. Podstawowa tablica dzieli mniejsze zadania na kolumny: zaplanowane, w trakcie, skończone, dodatkowo podzielić ją można na wiersze, gdzie każdy wiersz grupuje trochę inną część projektu. W przypadku tej pracy powstały kategorie: rzeczy do rozpoznania, praca pisemna, materiały, programowanie, konfiguracja projektu.

Literatura

[1] Creative Commons cc0 1.0 universal (cc0 1.0) przekazanie do domeny publicznej. https://creativecommons.org/publicdomain/zero/1.0/deed.pl. Data uzyskania dostępu: 2022-08-10.

[2] Material maker. https://godotengine.org/showcase/material-maker/. Data uzyskania dostępu: 2022-08-10.

[3] Oracle oracle jd edwards enterpriseone applications documentation, release 9.1.x. https://docs.oracle.com/cd/E16582_01/doc.91/e15086/und_configurator.htm#E0ABC00002. Data uzyskania dostępu: 2022-11-10.

[4] PWN internetowy słownik pwn. https://sjp.pwn.pl/slowniki/interakcja. Data uzyskania dostępu: 2022-09-10.

[5] Unreal Engine unreal engine 5.0 release notes. https://docs.unrealengine.com/5.0/en-US/unreal-engine-5.0-release-notes. Data uzyskania dostępu: 2022-07-10.

[6] Hubert Woźniak Andrzej Barczak. Comparative study on game engines. Technical report, Siedlce University of Natural Sciences and Humanities, Faculty of Exact and Natural Sciences, Institute of Computer Science, 2019.

[7] Dirk Bartz Bernhard Preim. *Visualization in medicine*. Morgan Kaufmann Publishers, 2007.

[8] Noble Joshua. *Programming Interactivity*. O'Reilly Media, Inc., 2012.

[9] Solomon Press Richard S. Gallagher. *Computer Visualization: Graphics Techniques for Engineering and Scientific Analysis*. CRC Press, CRC Press LLC, 12/22/94.

[10] Dodsworth Simon. *The Fundamentals of Interior Design*. AVA Publishing S.A., 2009.

[11] Boeykens Stefan. *Unity for Architectural Visualization*. Packt Publishing Ltd., 2013.

[12] Colin Ware. *Information Visualization Perception for Design*. Morgan Kaufmann Publishers, 2004.

Spis rysunków