

**Wydział**

**Nazwa Katedry**

Nazwa specjalizacji

**Imię Nazwisko**

Nr albumu xxxx

**Tytuł pracy dyplomowej**

Rodzaj pracy (np. licencjacka, inżynierska, magisterska, projektowa)

Imię i nazwisko promotora

miejsce, miesiąc, rok obrony

# Wstęp

Prace nad optymalizacją liczenia i wyświetlania scen i obrazów przygotowanych w trójwymiarze prowadzone są od lat. Nie chodzi tylko o film czy efekty 3D, ale również wykorzystanie grafiki 3D w sieci internetowej. Ich wykorzystanie początkowo głównie interesowało twórców gier, ale również osoby zainteresowane budowaniem wirtualnej rzeczywistości w sieci. Klasycznym już przykładem jest system wirtualnych światów *Second Life*, okrzyknięty fenomenem 2007 roku [1]. Jego popularność wiązała się w panującym wówczas trendem na *życie w sieci* ­– „(…) ludzie grupowo wynajmowali wyspy *Second Life*, próbując skolonizować cyberprzestrzeń, która nigdy się nie zmaterializowała[[1]](#footnote-1)”. 3D w sieci wówczas było namiastką wirtualnej rzeczywistości. Dzisiaj, kiedy mamy Oculus Rift czy Microsoft Hololens, wirtualna rzeczywistość nabrała innego, bardziej *realnego* znaczenia. Grafika 3D jest teraz narzędziem, a nie samą przestrzenią działania. W związku z powyższym, przykłady jej wykorzystania są bardziej abstrakcyjne i kreatywne, niż rekreacja istniejącej przestrzeni i postaci.

W projektowaniu interfejsów użytkownika sieci internetowej poprzez „użycie 3D” lub „efekty 3D” rozumie się dzisiaj wykorzystanie języka WebGL, standardu tworzenia grafiki trójwymiarowej przy wsparciu akceleracji sprzętowej. Standard ten zostanie szerzej opisany w dalszej części tej pracy. W tym momencie warto jednak wskazać, że wykorzystanie WebGL nie oznacza jedynie możliwości kreowania scen 3D z modelami, ale również rozmaite efekty wizualne wykorzystywane w nietypowych przejściach stron czy po najechaniu kursorem na obiekt. W tym standardzie możliwości wizualne do uzyskania w sieci są niemalże nieograniczone, bo o ile twórcy stron www dalej muszą się liczyć z ilością danych do pobrania przez klienta albo możliwościami przeglądarek, tak wsparcie sprzętowe wykonywanych obliczeń daje niesamowitą moc i przestrzeń kreacji.

# Cel i założenia projektu

Celem tej pracy jest przedstawienie możliwości, jakie dostarcza WebGL w projektowaniu i tworzeniu witryn internetowych. Przedmiotem pracy nie jest jedynie prezentacja potencjalnych pomysłów wykorzystania tego standardu, ale przede wszystkim zarysowanie schematów pracy nad tworzeniem tego typu efektów począwszy od modelowania 3D w dedykowanym programie aż po implementację w kodzie JavaScript.

Założenia projektu są następujące:

1. Projekt jest stroną www stworzoną do korzystania przez dowolnego użytkownika sieci
2. Projekt jest przykładem tzw. „wizualnej opowieści” (ang. visual storytelling), przedstawia konkretną historię lub raczej prowadzi przez zbiór informacji niczym film animowany
3. Przy przejściu do kolejnych zagadnień projekt wymaga interakcji z użytkownikiem. Wymóg interakcji jest jawnie zakomunikowany użytkownikowi lub wynika ze standardowych zachowań przy korzystaniu z witryn sieci web
4. Projekt pokazuje rozmaite przykłady wykorzystania standardu WebGL oraz grafiki 3D w prezentowaniu skomplikowanych ilustracji wspomagających przekazanie informacji

W niniejszym tekście zostanie dokładnie przedstawiona metodyka pracy wykorzystana w celu spełnienia wyżej określonych założeń projektu. Treść jest podzielona na trzy części, z których pierwsze dwie stanowią wprowadzenie do trzeciej. Najpierw zostanie naszkicowany temat grafiki 3D, software’u 3D oraz narzędzi, których wykorzystanie miało miejsce przy realizacji projektu. W drugiej części przedstawione zostanie środowisko programistyczne do budowania dobrze funkcjonującej aplikacji webowej, opisane standardy pracy oraz charakterystyka narzędzi. Trzecią częścią będzie spięcie obu powyższych i szczegółowe opisanie procesów implementacji grafiki 3D w kodzie strony internetowej.

## Wykorzystany software

### Grafika 3D

1. Maya 2018, Autodesk (*Student Licence*) – niesamowicie złożony, kompleksowy program do modelowania, animacji i renderowania grafiki 3D. Opanowanie programu Maya na poziomie podstawowym zajęło prawie trzy semestry przedmiotu specjalizacyjnego na kierunku informatyka. Maya, poza modelowaniem brył trójwymiarowych oraz animowaniem ich kształtów posiada rozbudowany silnik fizyczny, umożliwiający tworzenie wymyślnych symulacji.
2. Blender 2.82a – całkowicie darmowy program o możliwościach zbliżonych do programu Maya, jednak o zupełnie innym interfejsie.
3. FBX Review, Autodesk – niewielki program stworzony przez firmę Autodesk to podglądu i testowania plików wyeksportowanych z programów graficznych z rozszerzeniem .fbx

### Animacja i postprodukcja

1. Davinci Resolve 16, Blackmagic Design – potężny, darmowy program do montażu, edycji oraz nagrywania wideo. Rozszerzenie Fusion do tego programu dodatkowo umożliwia import brył 3D oraz animacji 3D, po czym daje narzędzia do edycji tych animacji czy ich koloryzacji lub montażu. Rozbudowana biblioteka efektów pozwala wzmacniać efekt pracy w programach 3D

### Implementacja

1. Visual Studio Code, Microsoft – darmowy program do edycji kodu. Posiada wbudowaną konsolę wiersza poleceń oraz szereg rozszerzeń, przykładowo do bieżącego sprawdzania poprawności semantycznej pisanego kodu.
2. Chrome, Google – darmowa przeglądarka od Google, która w swoich nowszych wersjach zapewnia bezproblemowe wykorzystanie standardu WebGL. Jest dodatkowo wyposażona w Chrome Developer Tools, czyli zestaw narzędzi deweloperskich, które pomagają w łatwy sposób wykryć błędy w kodzie.
3. Git, Git Foundation – system kontroli wersji kodu.

## Wykorzystane języki programowania i biblioteki

### Języki programowania i znacznikowe

1. JavaScript – to „skryptowy (interpretowany lub kompilowany metodą JIT) język programowania, w którym funkcje są *obywatelami pierwszej kategorii* – obiektami, które można przechowywać w zmiennych jako referencje i przekazywać jak każde inne obiekty”[[2]](#footnote-2) – to cytat za wstępem do dokumentacji JavaScript Mozilla Developer Network, jednej z najpełniejszych dokumentacji tego języka. JavaScript nie należy mylić z Javą, jedyny prawdziwy związek tych dwóch języków to to, że obydwa są znakami towarowymi firmy Oracle[[3]](#footnote-3). JavaScript wykorzystuje się w programowaniu aplikacji webowych, gdyż kompilator tego języka jest standardem każdej przeglądarki. JavaScript, jak według powyższego cytatu, opiera się na wykorzystaniu funkcji jako podstawowej jednostki.
2. CSS3 – system znacznikowy do zapisu styli obowiązujących na stronie www. CSS to z języka angielskiego kaskadowe arkusze styli (*cascade style sheets*). Uszczegóławiając: style to właściwości wyświetlanych elementów, typu pozycja, kolor, margines. Arkusz to inna nazwa pliku ze znacznikami i odpowiadającymi im parametrami i ich właściwościami. Kaskadowy oznacza priorytetyzację wykonania konkretnych zapisów. Jeśli coś jest zapisane wyżej w arkuszu, może zostać nadpisane wyżej. O priorytecie zapisu decyduje również szczegółowość znacznika. CSS3 to najnowszy standard CSS obsługujący również wydajne animacje elementów, również w przestrzeni 3D.
3. HTML5 – język znacznikowy wykorzystywany do zapisu układu elementów na stronie oraz ich wstępnej charakterystyki. W HTML są elementy odpowiedzialne za przechowywanie tekstu, obrazu lub zbiorów innych elementów. Poprawna semantyka HTML jest ważna zwłaszcza dla silników wyszukiwarek indeksujących witryny dla tysięcy zapytań dziennie. HTML5 to najnowszy standard HTML obsługujący m.in. dodatkowe zapisy ułatwiające korzystanie z sieci dla osób z niepełnosprawnościami oraz wydajniejszą obsługę materiałów multimedialnych, takich jak wideo, ale również obraz generowany bezpośrednio w kodzie strony internetowej.
4. GLSL – język skryptowy do zapisu operacji wykonywanych przy akceleracji sprzętowej karty graficznej maszyny, na której wyświetlana jest strona. GLSL (*graphic library scripting language*) jest wykorzystywany przy tworzeniu shader’ów (szerzej opisanych w dalszej części pracy).

### Biblioteki

1. Three.JS – biblioteka oparta o standard WebGL. Za pomocą Three można pominąć trud uczenia się WebGL, który ma dosyć skomplikowaną składnię oraz niską możliwość kontroli błędów w środowisku przeglądarki i posługiwać się tym językiem przy pomocy funkcji JavaScript. Three pozwala w łatwy sposób budować sceny 3D oraz animacje. Twórcy Three opracowali bardzo szczegółową dokumentację biblioteki opatrzoną przykładami, z których kilka jest zreprodukowanych w tej pracy.
2. PIXI.js – biblioteka również oparta o standard WebGL, jednak z mniejszym ukierunkowaniem na budowanie scen i animacji 3D jak Three, a większym na proste tworzenie efektów wizualnych opartych o shader’y.
3. GSAP, GreenSock – biblioteka do pracy z animacją na stronie. GSAP zawiera szereg narzędzi regulujących czas i kolejność animacji, z poszanowaniem standardów różnych przeglądarek. Dodatkowo zapewnia jak najsprawniejsze działanie animacji przez wykorzystanie do ich obsługi transformacji CSS, które są najwydajniejsze do tego typu operacji.
4. ScrollMagic.js – biblioteka pozwalająca skutecznie i bezbłędnie monitorować przesuwanie się po stronie przez użytkownika i na bazie informacji o jego obecnej pozycji wyzwalać animacje lub pokazywać elementy.
5. React.js – potężna biblioteka autorstwa firmy Facebook umożliwiająca budowanie szybkich i wydajnych aplikacji typu SPA (ang *single-page application*, aplikacja jednostronicowa) w oparciu o moduły. React opisany będzie szeroko w drugiej części dotyczącej realizacji projektu.

# Grafika trójwymiarowa – krótka charakterystyka

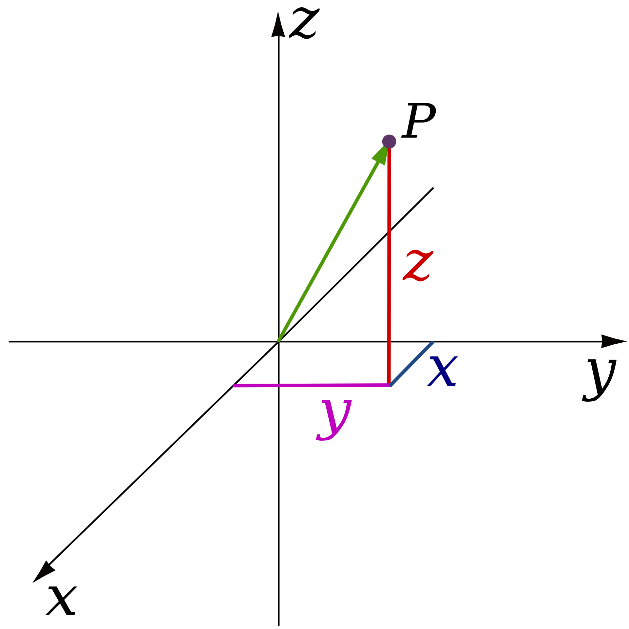
### Wprowadzenie

Dokładne zagłębianie się w charakterystykę matematyczną grafiki trójwymiarowej nie jest przedmiotem tej pracy. Należy jednak dla spójności przekazu wyjaśnić kilka podstawowych pojęć oraz nomenklaturę, która jest używana w tym tekście. Cytując za przytoczoną wcześniej książką Tony’ego Parisi[[4]](#footnote-4):

Komputerowa grafika trójwymiarowa (w odróżnieniu od dwuwymiarowej) to grafika wykorzystująca trzy wymiary do reprezentacji danych geometrycznych (często kartezjańskich), które są przechowywane w komputerze w celu wykonywania obliczeń i renderowania obrazów dwuwymiarowych. Obrazy takie można przechowywać w celu wyświetlenia w odpowiednim momencie albo wyświetlać na bieżąco.

Pozwolę sobie, podobnie jak autor cytowanego tekstu, rozbić tę definicję i objaśnić jej poszczególne elementy.

Co to znaczy, że grafika wykorzystuje trzy wymiary? W dużym skrócie można to wyjaśnić w ten sposób, że każdy z punktów na płaszczyźnie w trójwymiarze jest zapisywany przy użyciu trzech współrzędnych. Powyższe definicja przywołuje układ kartezjański i najczęściej spotkamy się z takowym podczas pracy z grafiką 3D:



Rysunek 1. Kartezjański układ trzech współrzędnych[[5]](#footnote-5)

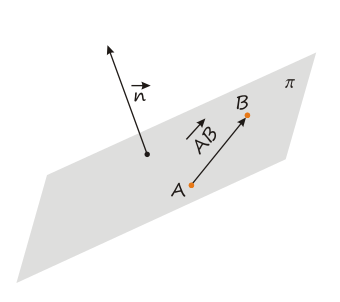
Płaszczyzna, do której należy punkt, może być albo częścią sceny albo jedną ze ścian bryły w scenie trójwymiarowej. W matematyce modeli trójwymiarowych nie jest zapisywany jednak każdy punkt na każdej płaszczyźnie. Przechowuje się współrzędne tych punktów, które są końcami odcinków między którymi rozpostarta jest płaszczyzna.

Inną ważną własnością płaszczyzn, o której będzie mowa w tej pracy, jest wektor normalny. Wektorem normalnym jest wektor prostopadły do płaszczyzny. Łatwa do zrozumienia definicja wektora normalnego płaszczyzny znajduje się w e-podręczniku do matematyki AGH w Krakowie[[6]](#footnote-6):

Wektor n⃗ jest prostopadły do płaszczyzny π, jeżeli dla dowolnych dwóch jej punktów A i B wektory AB→ oraz n⃗ są prostopadłe.

Każdy niezerowy wektor prostopadły do płaszczyzny nazywamy wektorem normalnym tej płaszczyzny.

Ilustruje to poniższa rycina:



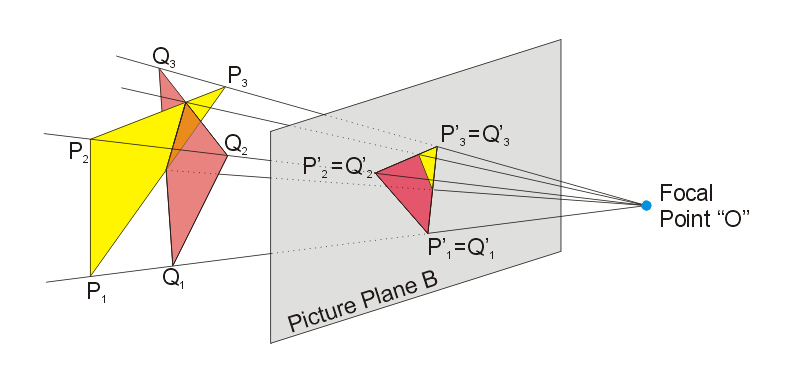
Rysunek 2 Wektor normalny[[7]](#footnote-7)

Wektory normalne płaszczyzn wykorzystuje się w np. w obliczeniach dotyczących ilości światła padającego na płaszczyznę i odbitego, przesłaniania się wzajemnego brył oraz wielu innych.

Posiadając takie podstawowe informacje jasne jest, że to właśnie między innymi te własności modeli 3D są wykorzystywane do obliczeń ich wzajemnego położenia, kolorów czy jasności. W tym miejscu istotne jest wyjaśnienie pojęcia renderowania.

Renderowanie jest kalką z języka angielskiego i nie ma swojego polskiego odpowiednika. W swoim ogólnym znaczeniu renderowanie to interpretacja treści elektronicznych w celu prezentacji ich w sposób odpowiedni dla swojego środowiska przeznaczenia. Przykładowo, podczas pisania tej pracy w programie Microsoft Word widok „kartki papieru” na której piszę jest renderem na ekranie jakiegoś zapisu cyfrowego o białym prostokącie.

W temacie grafiki trójwymiarowej, kontynuując objaśnianie definicji, renderowanie oznacza interpretację obrazu trójwymiarowego na finalnym dwuwymiarowym obszarze wyświetlania. Dwuwymiarowy zapis obrazu sceny 3D to zapis jej projekcji na płaszczyznę w stosunku do centralnego punktu wzrokowego. W dużym skrócie celem jest zarejestrowanie tylko tych fragmentów, które „widać” z danego kąta obserwacji sceny. Szczegółowe objaśnienie procesu renderowania nie jest przedmiotem tej pracy, ale rycina poniżej pozwala zrozumieć podstawowy problem do rozwiązania w algorytmach obsługujących projekcję, tj. zapis w przestrzeni dwuwymiarowej tylko tych punktów należących do danych brył, które mogą zostać wyświetlone.



Rysunek 3 Projekcja kompozycji 3D na płaszczyznę B[[8]](#footnote-8)

Kolejnym problemem renderowania, który na tej rycinie jest już rozwiązany, ale jest mocno skomplikowany, jest problem światła i cienia, wzajemnego oświetlenia brył oraz ich ustawienia względem oka. Renderer musi oszacować, który element znajduje się bliżej oka, a który dalej, zapis samych punktów może nie wystarczyć.

Wracając do definicji grafiki trójwymiarowej należy jeszcze przyjrzeć się ostatniemu zdaniu. Przechowanie obrazu do wyświetlenia w odpowiednim momencie jest dosyć jasne, przykładowo może chodzić o klatki animacji w filmie 3D. Wyświetlanie na bieżąco jest tematem bardziej skomplikowanym. Powyżej, bardzo pobieżnie, zostały opisane matematyczne zależności dotyczące bryły 3D. W zasadzie pominęłam kwestie oświetlenia i teksturowania, które, wykonane poprawnie, nadają bryłom realizmu. Jak zapewnić, by ta mnogość operacji była w stanie wykonać się tak szybko, by móc wyświetlać animację 3D na bieżąco? Ten temat będzie się pojawiał przy objaśnianiu kolejnych przykładów produkcji scen 3D w programie, a także przy użyciu WebGL.

### Nazwy i pojęcia

W tej pracy pojawi się wiele nazw, które należy objaśnić przez rozpoczęciem korzystania z nich. Wiele z nich ma swoje odpowiedniki w języku polskim (podane poniżej kursywą), ale w użyciu profesjonalnym używa się dla wygody nazw angielskich i mogę z nich korzystać wymiennie w dalszych partiach tego tekstu. Mogą też pojawiać się na rycinach, w kodzie lub w zdjęciach interfejsu programów graficznych.

1. Vertex (*wierzchołek*) – punkt w przestrzeni 3D. Pomiędzy wierzchołkami rozpięte są odcinki, które najczęściej wyznaczają brzegi płaszczyzny będącej częścią składową bryły. Wierzchołki mogą mieć też obiekty dwuwymiarowe w przestrzeni 3D np. krzywe.
2. Edge (*brzeg*) – odcinek wyznaczony przez dwa wierzchołki. Pomiędzy odcinkami może rozpięta być płaszczyzna
3. Face (*ściana*) – płaszczyzna, będąca częścią składową bryły 3D. Ściany zawsze są wielokątami, najczęściej czworokątami lub trójkątami
4. Mesh (*siatka*) – dwie lub więcej ściany połączone ze sobą krawędziami. Siatka jest opisem bryły (modelu) w przestrzeni trójwymiarowej
5. Geometria – zbiór danych dotyczących wierzchołków, wektorów normalnych, pozycji bryły.
6. Materiał – wygląd ściany bryły, tak naprawdę zapis właściwości, jakie musi wziąć pod uwagę renderer, interpretując daną bryłę, przykładowo: kolor, ilość odbijanego światła, refleksy etc.
7. Tekstura –obraz (ze zdjęcia, rysunek, wygenerowany cyfrowo) nakładany na materiał w celu uzyskania konkretnych efektów wizualnych
8. Bump map (*mapa odbić*)
9. Normal map (*mapa wektorów normalnych*)
10. Displacement map (*mapa przesunięć*)
11. Scena – zbiór elementów 3D oddziałujących na siebie wzajemnie
12. Kamera – punkt widzenia na scenę 3D, wykorzystywany przy renderowaniu
13. Renderer – program lub funkcja interpretująca scenę 3D na obraz 2D

# Charakterystyka projektu graficznego

## Dziedzina problemowa

Warto stanąć w korytarzu dowolnego centrum handlowego, najlepiej w drugiej połowie dnia w weekend i zwrócić uwagę na krążące wokół tłumy. Następnie wejść do dowolnego sklepu, najlepiej odzieżowego i spojrzeć na uginające się od towarów półki i wieszaki. Później, zanim dostanie się bólu głowy od hałasu i nadmiaru bodźców wzrokowych, zadać sobie kilka pytań, wszystkie zaczynające się od: ile? Ile trzeba wyprodukować obiektów, aby przy takim popycie półki nadal wydawały się pełne? Ile różnych sklepów może sprzedawać ten sam produkt, nie tracąc na tym zysku? Ile ubrań, butów, rajstop i zegarków potrzebować może jedna osoba? Ile towarów na koniec miesiąca nie znajdzie nowych właścicieli?

Jeszcze przed rozpoczęciem researchu do pracy wiedziałam, że jest coś głęboko nieetycznego w t-shircie za 15 złotych w dwupaku z napisem CONSCIOUS (ang. świadomy, w domyśle chodzi o świadomość kosztu ekologicznego produkcji). Domyślałam się też, że jest coś patologicznego w tym, że ludzie wydają pieniądze, których nie mają na rzeczy, których nie potrzebują i które nawet im się nie podobają[[9]](#footnote-9).

Research do projektu wykonał się prawie sam. Zaczęłam od obejrzenia filmu True Cost z 2015 roku, a na bazie usłyszanych tam informacji wystarczyło wpisać odpowiednie hasła w wyszukiwarkę, by utonąć w morzu rekordów. Rynek tzw. “szybkiej mody” (ang. fast fashion) jest jednym z najgorszych w kwestii zanieczyszczania środowiska[[10]](#footnote-10).

## Wizualna opowieść

Skala problemu jest tak duża, a waga pozyskanych informacji tak ciężka, że pomysł zebrania tych liczb i haseł w całość, nadania im ekspresyjnej formy i udostępnienia jak największej liczbie odbiorców wydaje się czymś oczywistym. Wybrałam w tym celu formę „wizualnej opowieści” (ang. *visual story-telling*), gdyż z doświadczenia wiem, iż jest czymś bardziej pochłaniającym niż suche fakty przedstawione przy pomocy tekstu. Wizualna opowieść może być częścią filmu czy animacji, ale może też być częścią tekstu. W tym ostatnim podejściu nie chodzi o zdania opatrzone ilustracjami, ale raczej o obrazy z dodatkiem niewielkiej ilości słów dla podkreślenia swojej wagi.

W projekcie strony pojawiają się trzy części główne i wiele mniejszych. Główne partie to: wstęp, gdzie pojawia się kilka liczb ze statystyk, animowanych dla podkreślenia dramatyzmu. Wstęp ma przykuć uwagę odbiorcy do liczb pojawiających się w całej treści strony, nawet kiedy towarzyszące im obrazy będą znacznie ciekawsze. Kolejną partią są sekcje problemowe: ilość ubrań na osobę, masa produkcji branży modowej, wyzysk pracy na początku łańcucha dostaw, niska jakość tekstyliów, toksyny wykorzystywane przy pozyskiwaniu materiałów i ich wpływ na środowisko. Ostatnią partią jest sekcja źródeł i przydatnych linków, skąd odwiedzający może zaczerpnąć więcej wiedzy i uzyskać inspirację do bardziej świadomej konsumpcji. Ta część jest bardzo istotna, gdyż zdaję sobie sprawę, że przedstawienie poszczególnych zagadnień w sposób ciekawy wizualnie sprawia, że można przekazać tylko procent informacji, które są niezmiernie istotne w opisie tego problemu. Dodatkowo jest to swoisty podział na wstęp, rozwinięcie i zakończenie, jak w przykładnej opowieści przystało.

# Realizacja projektu

## Grafika 3D

### Modele ubrań

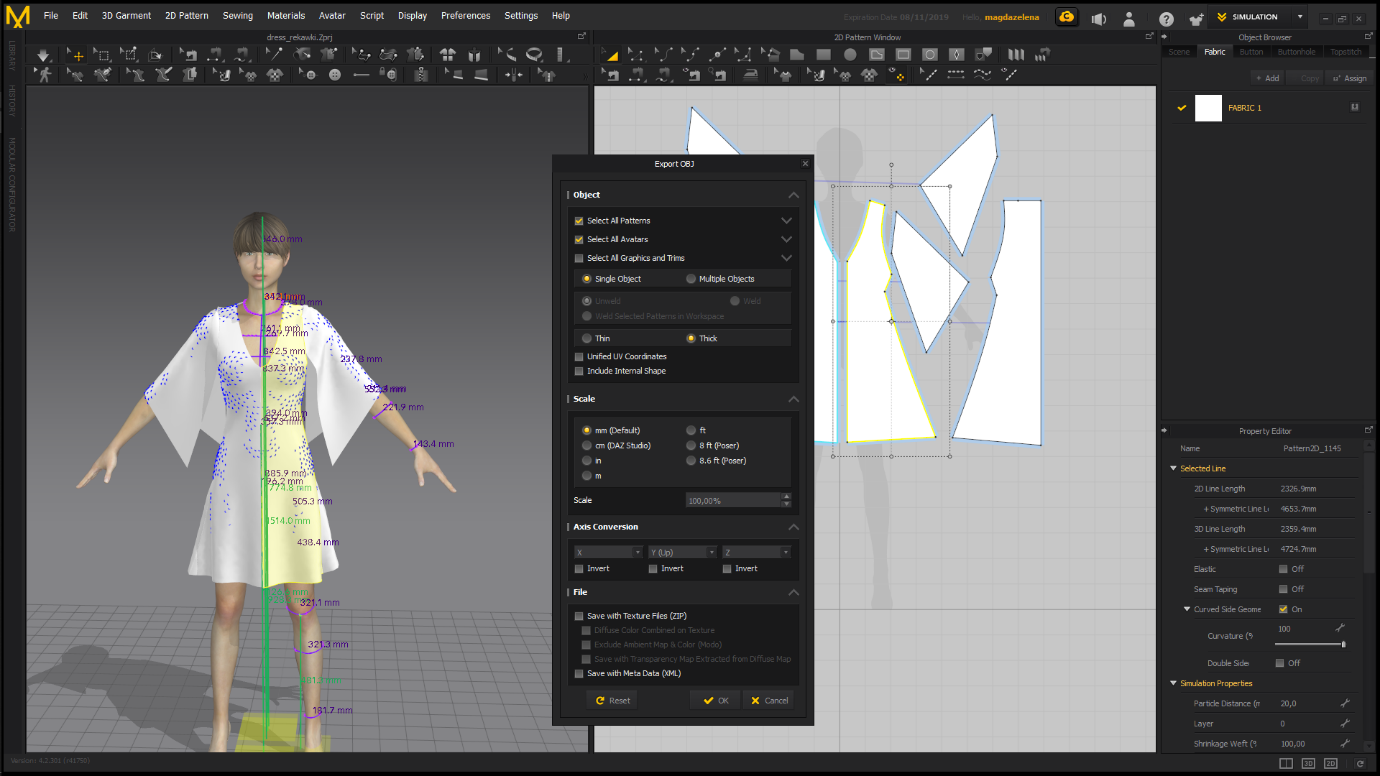
W projekcie pojawia się jedenaście różnych modeli sukienek. Sukienki są symbolami wszystkich ubrań ogólnie, wybrałam je jako przykład, gdyż to do kobiet najczęściej kierowane są reklamy, którymi bombardują nas sklepy.

Modele sukienek zostały wykonane przy pomocy programu Marvelous Designer 8. Jest to program dedykowany do modelowania wzorów odzienia. Praca z nim jest nieco odmienna od typowej pracy z modelowaniem 3D. Aby wykonać poprawną formę okrycia należy, niczym prawdziwa krawcowa, przygotować odpowiedni wykrój ubrania, dodać szwy (określić wcześniej ich rodzaj, elastyczność). Trzeba wybrać również rodzaj i strukturę materiału, czy ma być elastyczny czy sztywny, ciężki czy lekki. O ile interfejs programu jest bardzo intuicyjny, najdłużej zajęła mi nauka przygotowania poprawnych wykrojów ubrań.

Kolejność wykonywanych zadań w celu uzyskania poprawnego modelu w Marvelous Designer 8 jest następująca:

1. Przygotowanie wykroju ubrania
2. Umieszczenie elementów wykroju dookoła modelu awatara – przykładowo front bluzki z przodu klatki piersiowej
3. Na wykroju zaznaczenie brzegów, które mają zostać zszyte razem, a następnie dodanie w tym miejscu szwów
4. Po ukończonych przygotowaniach kliknięcie „play” nad obrazem awatara, które włącza „fizykę” – materiał opada w zgodzie z grawitacją, ale szwy ściągają połączone brzegi do siebie. Automatycznie buduje się uszyte ubranie
5. Fałdy materiału można przesuwać na modelu dokonując „poprawek stylistycznych”
6. Gotowe ubranie eksportuje się jako trójwymiarowy model do użycia w innych programach graficznych

Poniżej znajduje się zapis obrazu z przygotowania modelu jednej z sukienek:



Po prawej stronie widzimy ekran wykroju. U jego góry widoczne są niewielkie ikonki z narzędziami do tworzenia wielokątów, modelowania krzywych czy wycinania dziur. Nieco niżej znajdują się ikony maszyn do szycia, które odpowiedzialne są za dodawanie szwów we wcześniej zaznaczonych miejscach wykroju.

Lewa strona przedstawia obraz awatara z już nałożonym i zamodelowanym wykrojem sukienki. Niebieskie kropki na powierzchni materiału to miejsca styku ubrania z ciałem. Im większa gęstość kropek, tym ubranie jest ciaśniejsze na awatarze – pozwala to odpowiednie dobranie rozmiarów na wykroju. Na gotowej sukience oznaczone są też wszystkie miary ubrania oraz miary względem wysokości postaci – mamy dostępne awatary różnej płci, postury czy wieku.

Na środku widać okno ustawień do eksportu modelu do formatu .obj (Object 3D). Sukienkę eksportowałam jako Single Object, czyli pojedynczy obiekt. Można też wybrać Multiple Objects, kiedy zamodelowana jest np. bluzka i spódnica. Wyeksportują się wtedy jako dwie osobne bryły.

Postępując według wyżej opisanego schematu zadań stworzyłam jedenaście różnych modeli sukienek, które zaimportowałam w formacie .obj do programu Maya:



Rysunek 4 Modele sukienek zaimportowane do Maya 2018

### właściwości fizyczne tkanin

O ile zachowanie właściwości fizycznych tkanin jest podstawą działania programu Marvelous Designer i jest integralną częścią działania programu, w Maya oraz w innych środowiskach do pracy z 3D należy posiadać pewną wiedzę, aby osiągnąć zbliżony efekt.

Warto jednak zauważyć w tym miejscu, że choć tkanina jest de facto, materiałem, z którego wykonany jest jakiś obiekt, w pracy z grafiką 3D nie należy traktować tych dwóch pojęć jako tożsamych. Tak, jak to było wskazane powyżej w słowniku pojęć, materiał bryły 3D to zestaw cech face’ów niezbędnych przy renderowaniu. Materiał w tym kontekście przekaże informacje o kolorze, odbiciu światła etc. Materiał nie ma możliwości dodania cech fizycznych obiektowi, na który jest nałożony. O ile, przykładowo, będziemy chcieli imitować stal, możemy w materiale nadać ścianom srebrny kolor i wysoką połyskliwość, ale nie nadamy twardości czy trwałości czy wagi. Takie właściwości zaaplikujemy równolegle na sam mesh.

W Maya jest wygodne narzędzie do nadawania własności tkanin siatce obiektu. Nazywa się nCloth i, według definicji tłumaczonej z języka angielskiego z dokumentacji programu, jest to:

(..) szybkie i stabilne rozwiązanie do tkanin, które wykorzystuje system połączonych cząsteczek do symulowania szerokiej gamy dynamicznych powierzchni wielokątów. Na przykład, nCloth jest wystarczająco elastyczny do symulacji następujących substancji: ubrań tekstylnych, nadmuchiwanych balonów, powierzchni tłukących się oraz obiektów deformowanych[[11]](#footnote-11).

Rozszerzając nieco tę definicję, nCloth pozwala na dodanie obiektu do przestrzeni, w której oddziaływają na niego prawa fizyki, takie jak grawitacja oraz dodatkowo rozszerza jego własne właściwości o takie parametry jak waga, ciągliwość, elastyczność, trwałość etc. Dodatkowo symulator nałożony na siatkę sugeruje się jej pierwotnym kształtem, ale wpływa na poszczególne elementy siatki w celu uzyskania konkretnych efektów.

W projekcie użyłam nCloth do uzyskania dwóch efektów. Pierwszy z nich to eksplozja sukienki. Przygotowane wcześniej modele ubrań chciałam transformować jedno w drugie. Nie jest to jednak możliwe przy pomocy klasycznego przekształcania brył za pomocą Maya, gdyż siatki posiadają różną liczbę ścian. Modele są na tyle różne, że ręczne dopasowanie liczb face’ów do siebie nie wchodziło w grę, dodatkowo uzyskanie satysfakcjonujących rezultatów tą metodą mogłoby okazać się niemożliwe.

Innym rozwiązaniem jest zamiana jednego obiektu na drugi. Aby uzyskać efekt przejścia między modelami, pierwszy obiekt będzie niszczony do chmury małych fragmentów, z których następnie formować będzie się kolejny.

W pierwszym podejściu do realizacji tego pomysłu zaczęłam od „rozpadu” obiektu sukienki na cząsteczki - particles, przy użyciu generatora particles w Maya. Umieściłam słowo rozpad w cudzysłowie, gdyż tak naprawdę cząsteczki nie są częścią siatki, ale zupełnie oddzielnymi bytami. Taki pseudo-rozpad można uzyskać wybierając siatkę jako pierwotny kształt chmury cząstek, a sam materiał siatki ustawiając na niewidoczny. Efekty są ciekawe, ale w trakcie pracy sprawdziłam, że particles nie są możliwe do wyeksportowania jako animacja i moja kontrola nad nimi byłaby ograniczona. Generowanie ich przez JavaScript tą metodą do uzyskania satysfakcjonujących mnie efektów nie byłoby dostatecznie wydajne.

Drugie podejście zakładało faktyczny rozpad obiektu na małe cząstki - najlepiej po linii brzegowej face’ów. To rozwiązanie po pierwsze zapewniłoby mi możliwość eksportu animacji jako obiektu, a po drugie zachowałoby efekt operowania na bezpośrednio na tkaninie i nadało wrażenie realności. Właśnie ten proces opiszę w szczegółach w następnym podrozdziale.

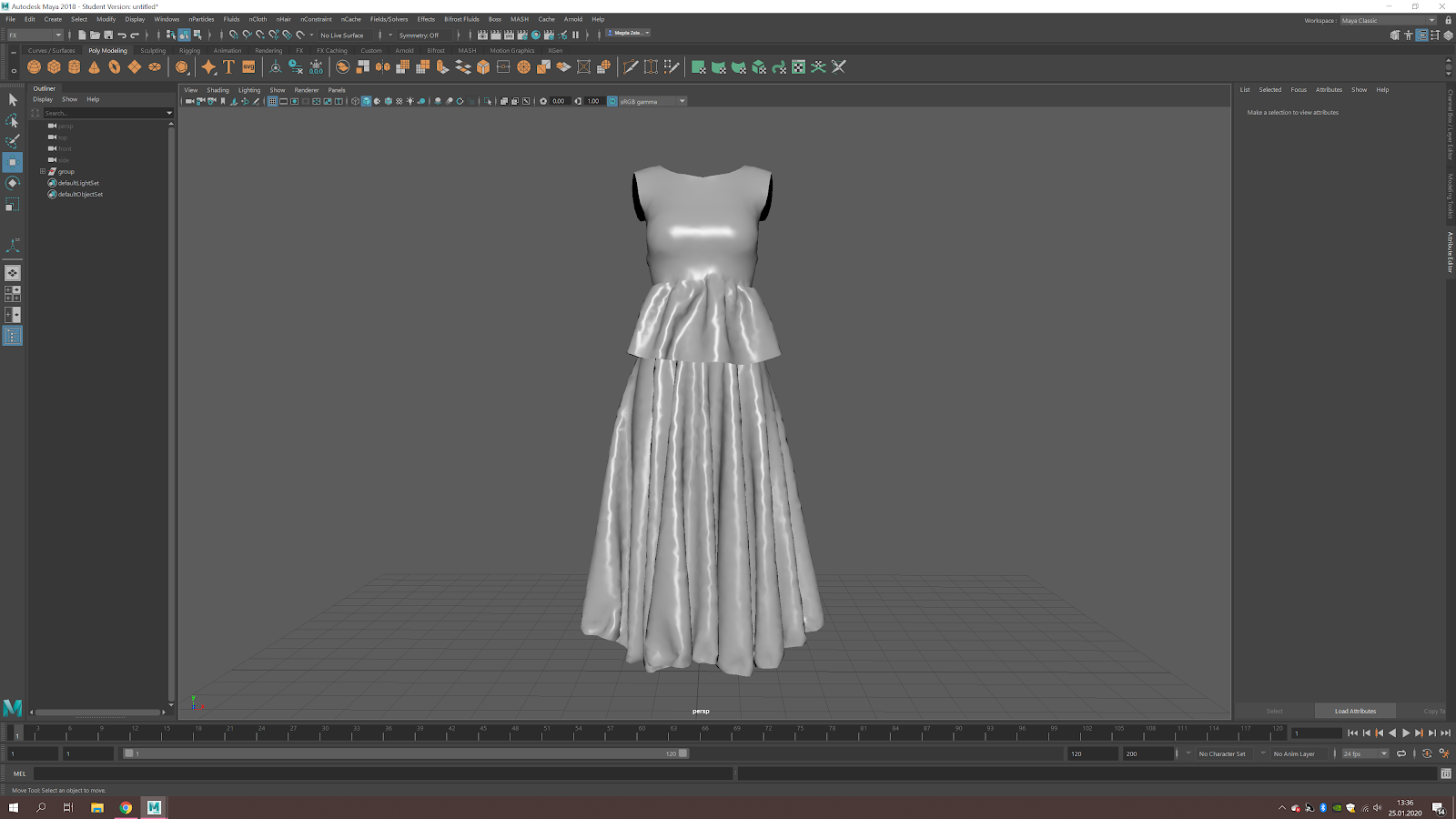
Drugi efekt do którego wykorzystałam nCloth to symulacja poruszania się ubrania bez postaci w środku, na niewidocznym wieszaku. Sam ten proces jest opisany poniżej i jest o wiele mniej skomplikowany niż eksplozja, ale trudności dostarczył eksport brył z zachowaniem animacji materiału, jednak bez silnika fizycznego.

### Eksplodowanie sukienki

Proces pracy był następujący:

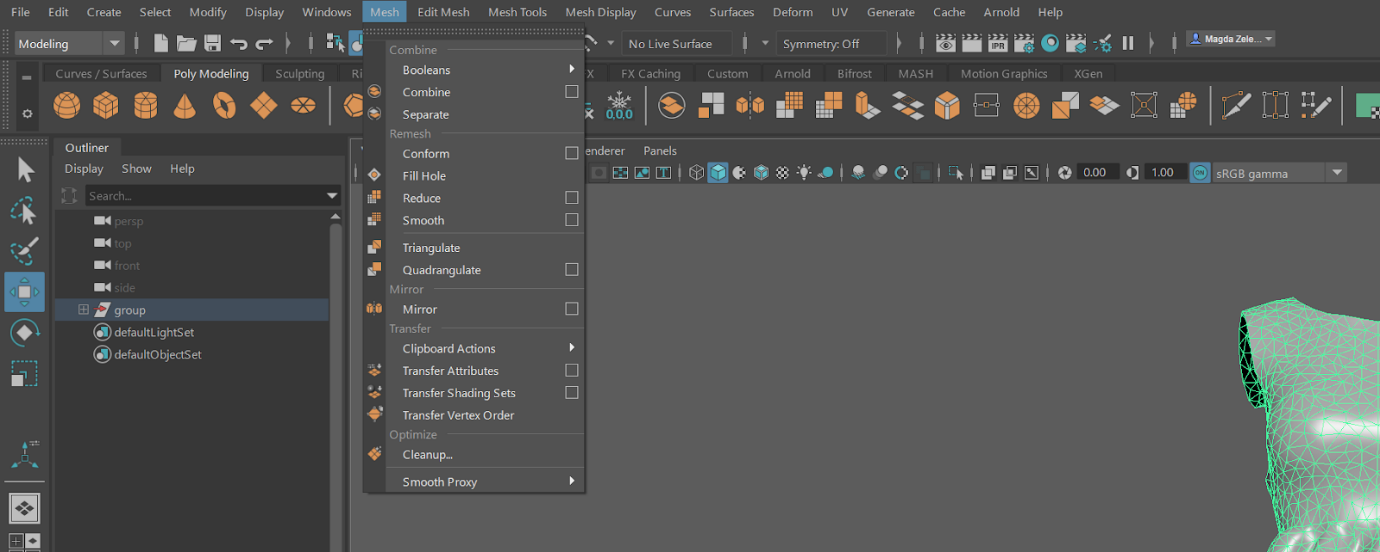
1. Obiekt otrzymuje cechy tkaniny nCloth, usuwam z otoczenia grawitację, by fizyka dotyczyła tylko struktury ścian sukienki,
2. Nadaję wszystkim brzegom face’ów charakter „rozrywanej powierzchni” - w razie kolizji z innym obiektem mam kontrolę nad tym, gdzie przerwie się materiał,
3. W środku obiektu umieszczam inny - niewielki prosty model geometryczny, jego tekstura będzie przeźroczysta, wobec czego sam model niewidoczny,
4. Bryle nadaję cechy obiektu pasywnego, a więc będzie miała cechu fizyczne i będzie mogła wchodzić w kolizję z innymi obiektami,
5. Animuję rozmiar bryły w czasie od niewielkiego do o wiele wykraczającego poza obszar modelu sukienki.

Dzięki nadaniu odpowiednich cech tkaniny modelowi sukienki, bryła powinna rozerwać go od środka. Ponieważ sukienki miały różne formy, do kolizji z nimi użyłam również różnego rodzaju brył. Poniżej przykład krok po kroku rozerwania materiału za pomocą bryły walca:

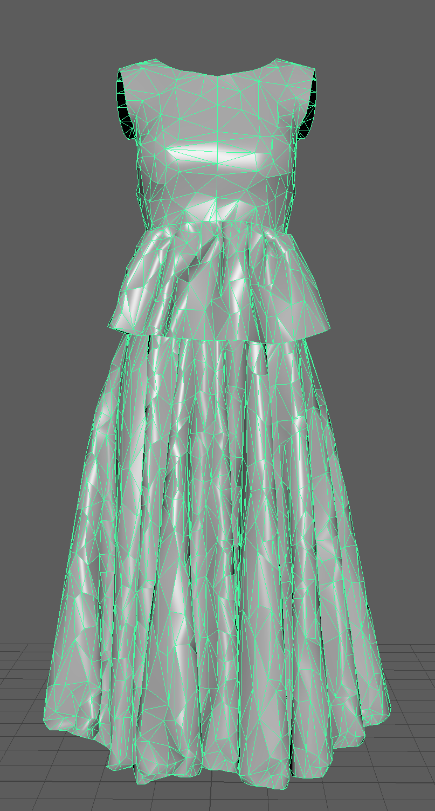
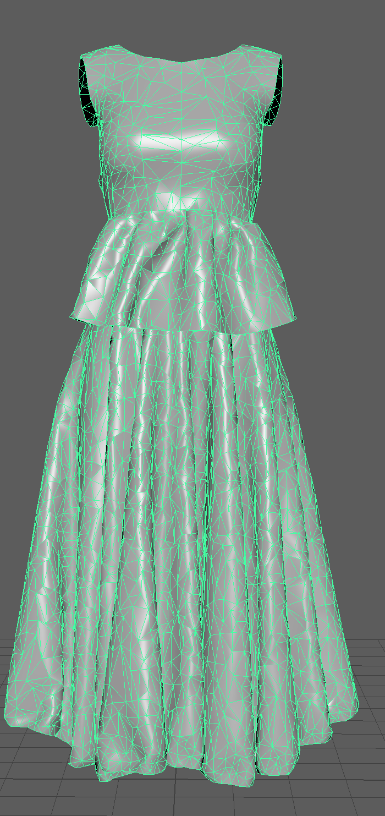
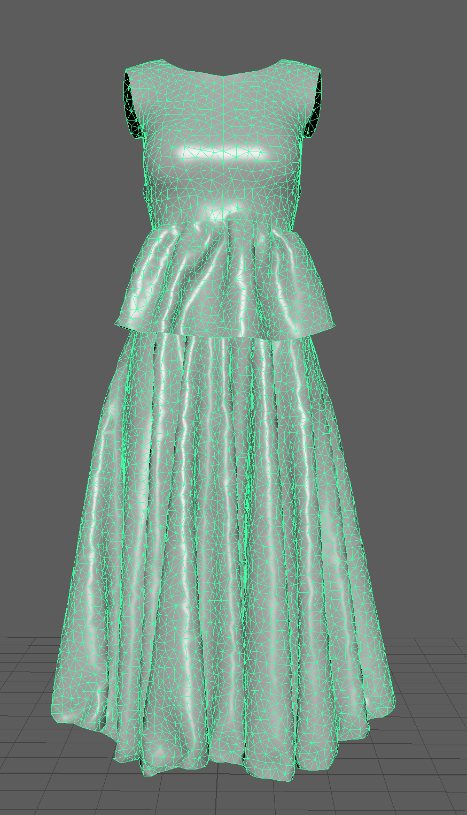


Rysunek 5 Model sukienki w programie Maya

Wykonany przy pomocy Marvelous Designer model sukienki umieszczam w nowej scenie w Maya. Model jest stosunkowo realistyczny, co oznacza, że składa się z ogromnej liczby face’ów. Obliczenie trajektorii lotu i deformacji dla każdej z nich będzie niezwykle kosztowne, więc należy pójść na kompromis i zrezygnować z części szczegółów na rzecz interesującego efektu. Korzystając z narzędzia Mesh->Reduce redukuję liczbę ścian modelu aż do uzyskania takiej formy, która jeszcze eksponuje część pierwotnych szczegółów, ale będzie optymalna do późniejszej animacji.

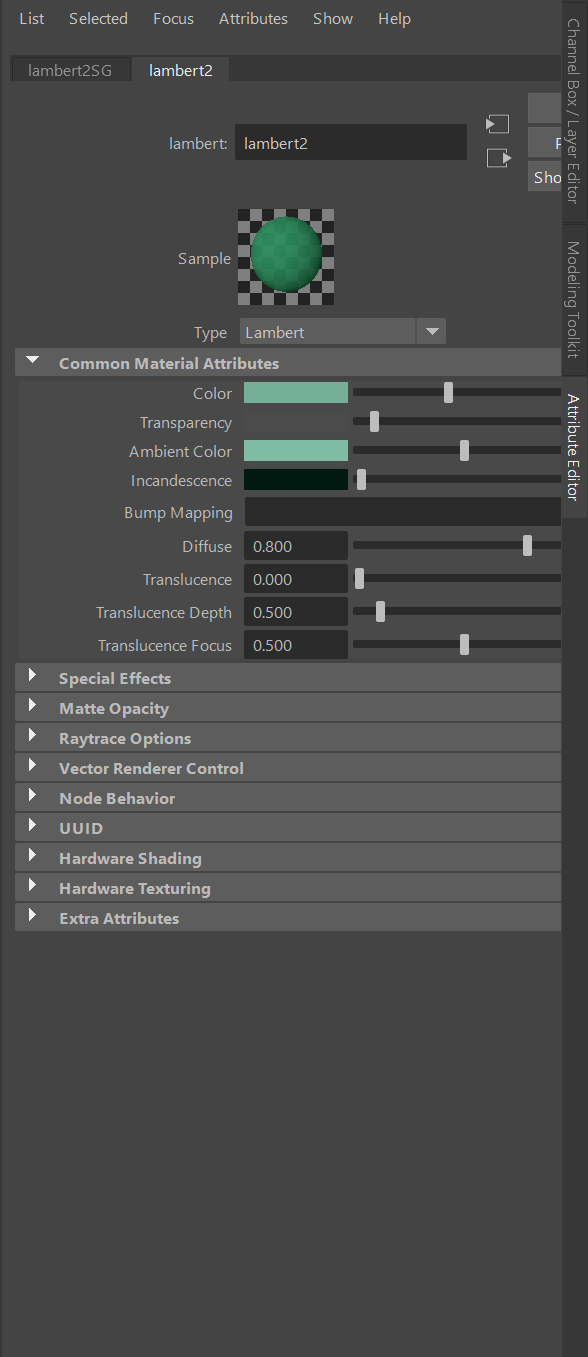
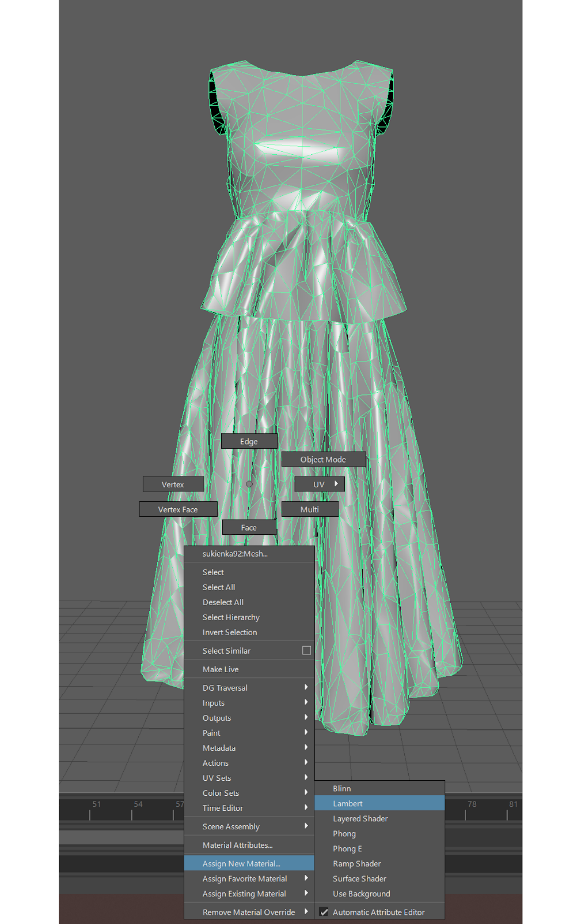


Rysunek 6 Opcja Mesh->Reduce znajduje się w menu Modeling



Rysunek 7 Wygląd siatki po kolejno wykonanych redukcjach

Niestety widać znaczną różnicę w jakości końcowej bryły. Dodanie odpowiedniej tekstury zredukuje nieco nieprzyjemny efekt kanciastości. Wybieram materiał Lambert, dzięki któremu łatwo uzyskam matową powierzchnię, a zależy mi na złagodzeniu kantów i odbić. Tekstura, którą dodaję, jest jasna, nieco przezroczysta, odbijająca światło. Załamania bryły, zwłaszcza na fałdach sukni stają się mniej widoczne, trochę rozmyte, ale zważywszy na fakt, że model będzie się poruszać, taki efekt jest jak najbardziej akceptowalny.

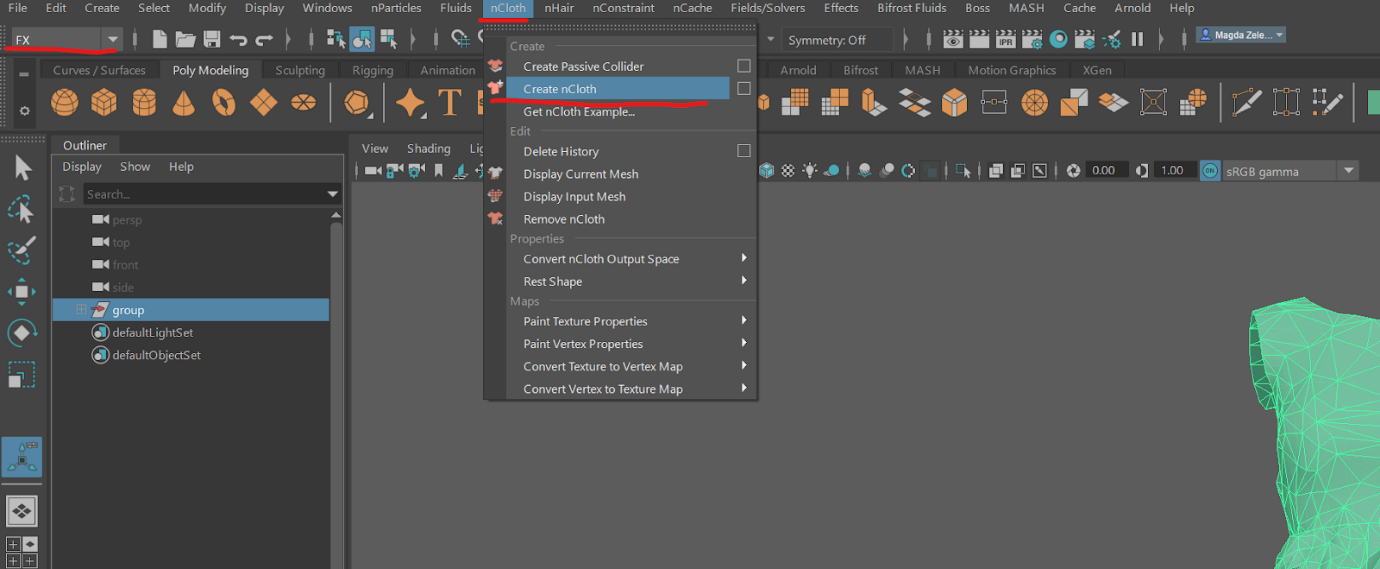


Rysunek 8 Proces dodania nowego materiału do siatki oraz przykład ustawień materiału

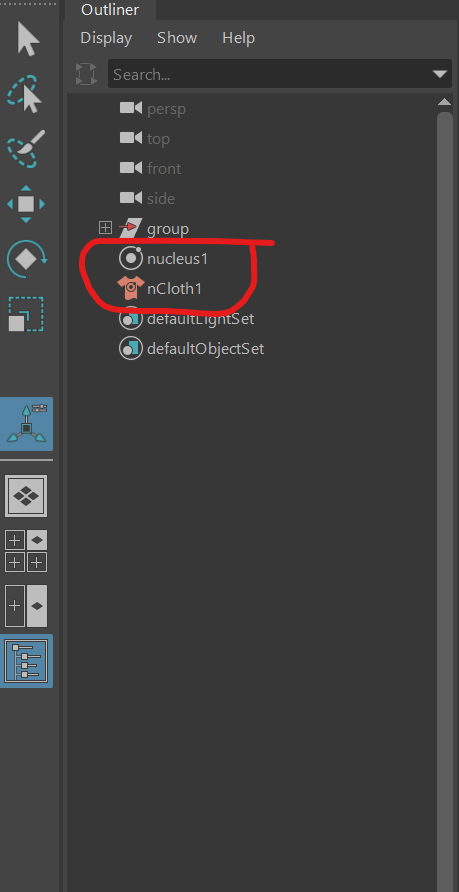


Rysunek 9 Model po redukcji liczby ścian i nadaniu materiału

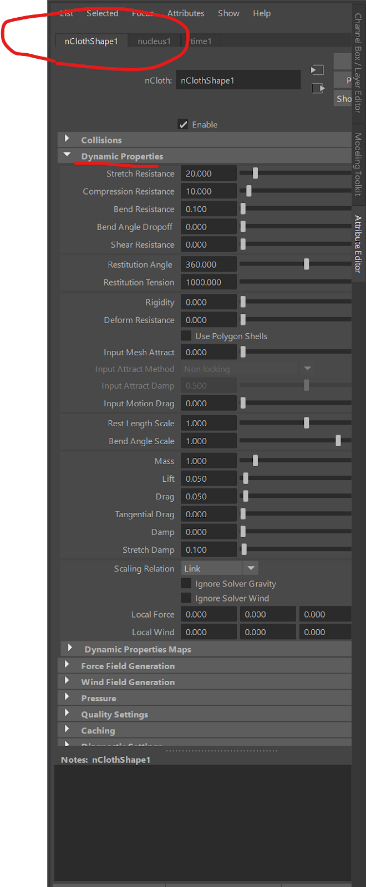
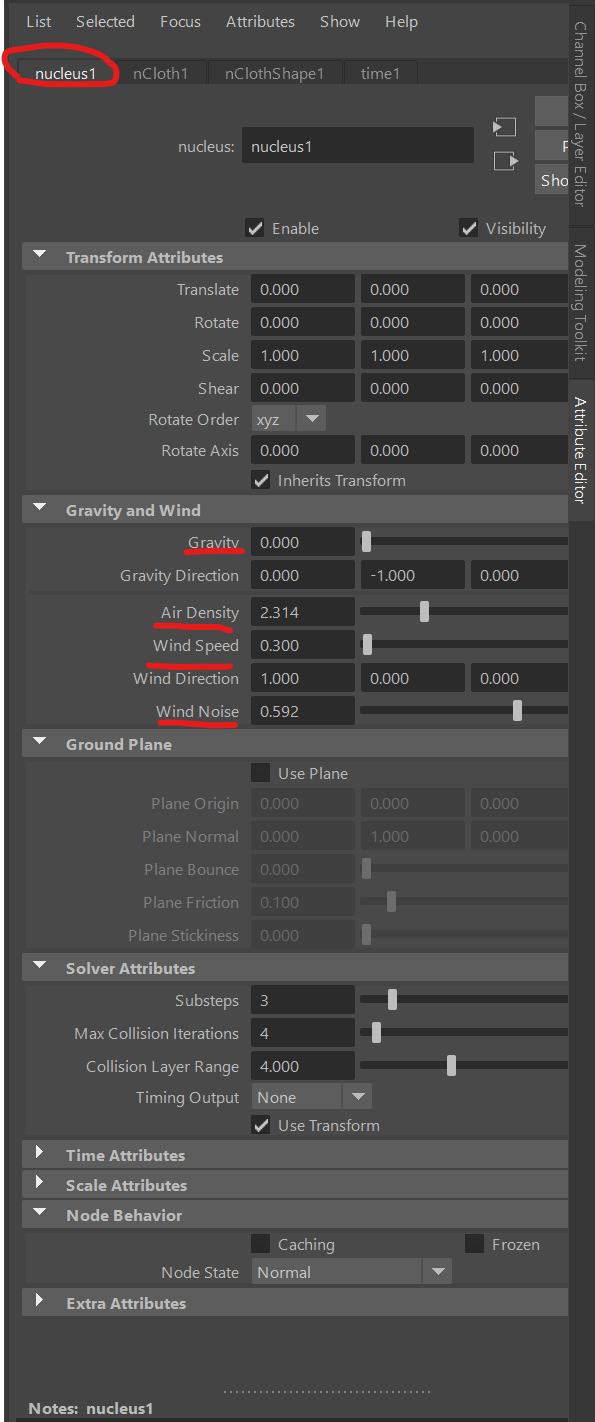
Następnie model potrzeba zmienić na materiał nCloth. W tym celu na pasku narzędzi przełączę się z menu modelowania na menu FX i wybiorę odpowiednią funkcję z menu nCloth:



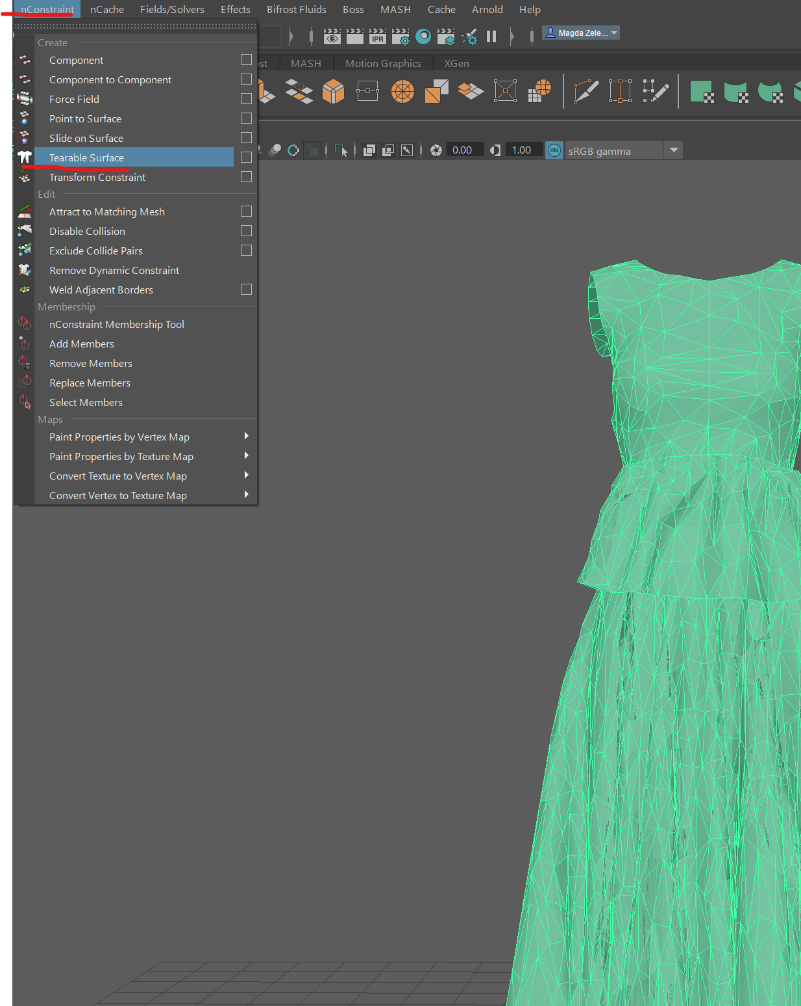
Rysunek 10 Utworzenie symulatora nCloth na modelu

Dodane zostały dwa nowe obiekty, takie jak nucleus i sam nCloth. Nucleus to węzeł pozwalający na wykonywanie matematycznych obliczeń związanych z fizyką innych węzłów, takich jak m.in. nCloth albo nParticles. Dzięki niemu można kontrolować takie właściwości jak np. wiatr czy grawitacja oddziaływująca na obiekt. Aby model sukienki nie opadał w dół, należy zmienić wartość siły grawitacji z domyślnej 9,8 na 0. Dodatkowo warto będzie wskazać na charakter materiału obiektu, więc można nadać niewielką wartość sile wiatru, aby fałdy sukienki delikatnie falowały. Cechy samego materiału można edytować w zakładce nClothShape wybierając właściwości dynamiczne (*Dynamic Properties).* Zwiększenie wartości odporności na rozciąganie i zginanie (odpowiednio *Streach Resistance* i *Bend Resistance*) sprawi, że sukienka rozerwie się, a nie rozciągnie na walcu jak guma. W celu osiągnięcia ciekawych efektów animacji można też nieznacznie zwiększyć wartość trwałości (*Rigidity)* o dosłownie 0.004 jednostki - wówczas po rozpadzie fragmenty będą walczyć z siłą eksplozji i dążyć do powrotu do swojej oryginalnej pozycji na modelu.

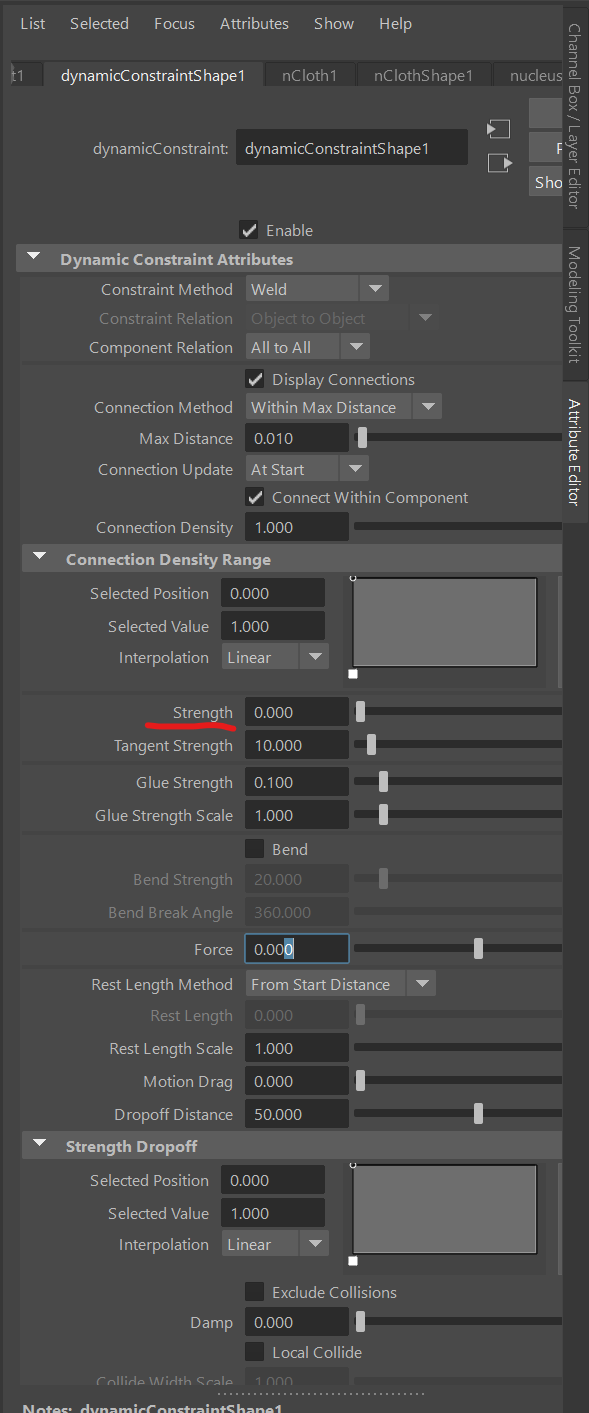
Rysunek 11 Obiekty po utworzeniu symulatora

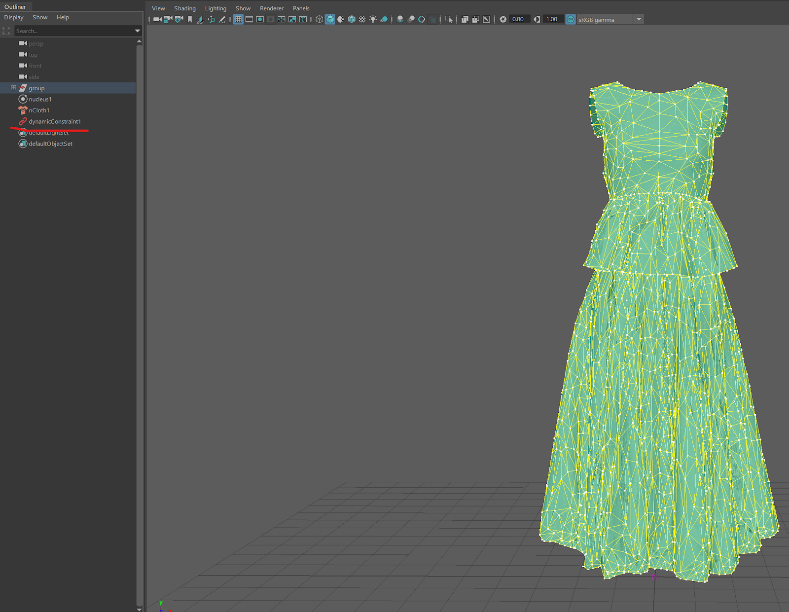


Rysunek 12 Ustawienia obiektów Nucleus i nCloth

Wartości obu węzłów można dostosowywać w trakcie testowania działającej animacji. Teraz należy zadbać, żeby model sukienki faktycznie rozerwał się na małe fragmenty. W tym celu należy zaznaczyć odpowiednie miejsca do przecięć, ustawiając nCloth jako powierzchnię rozrywalną (*Tearable Surface*). W tym celu, zaznaczywszy obiekt, z menu górnego wybieramy rodzinę węzłów nConstraint, a typ węzła Tearable Surface.

Po chwili obliczeń widzimy, że wszystkie brzegi face’ów modelu zmieniły kolor, a ich przecięcia są zaznaczone kropkami. Oznacza to, że wszystkie zmieniły się na szwy do przerwania. Oczywiście można nie wybierać wszystkich brzegów, ale wybrać tylko te interesujące, zaznaczając te po kolei. Nie jest to jednak celem tego przykładu.

Po utworzeniu węzła widzimy, że pojawił się on w menu po lewej stronie. Wybierając go, mamy dostęp do jego właściwości fizycznych.

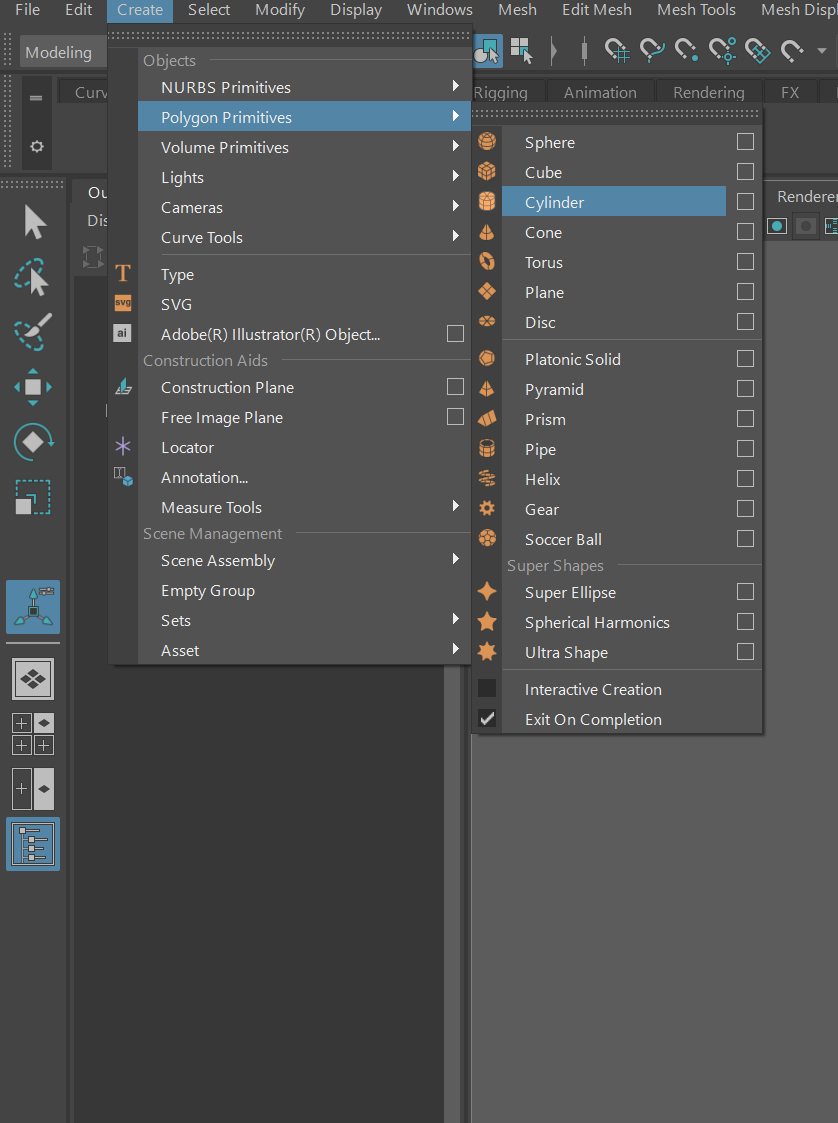


Rysunek 13 Model po dodaniu węzłów

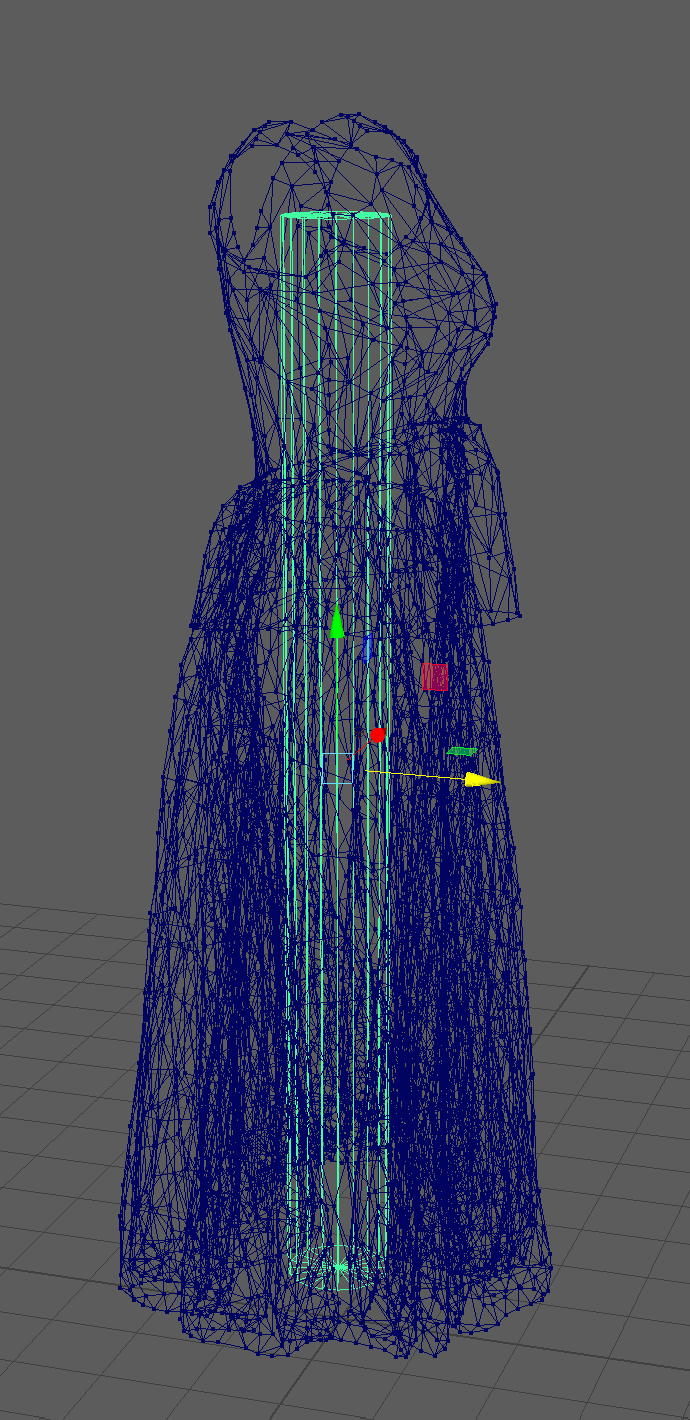
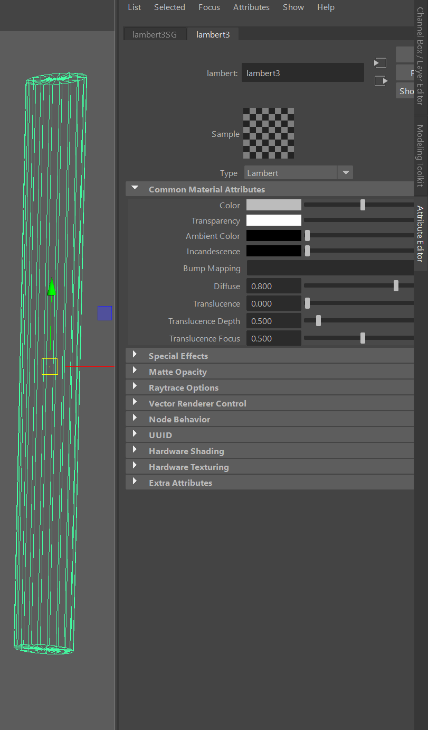
Rysunek 14 Ustawienie rozrywanej powierzchni na modelu

Manipulując wartością siły (*Strength*) trzymania szwów można ułatwić lub utrudnić darcie się tkaniny. W tym przykładzie wartość jest ustawiona na 0, aby zapewnić, że prawie każdy fragment będzie łatwo odchodził od drugiego pod wpływem siły uderzenia.

Jeśli właściwości fizyczne materiału są już gotowe do rozerwania, należy teraz stworzyć siłę, która na ten materiał zadziała. W tym celu w menu pomocniczym należy przejść z FX na Modeling i stworzyć nową bryłę typu np. walec.

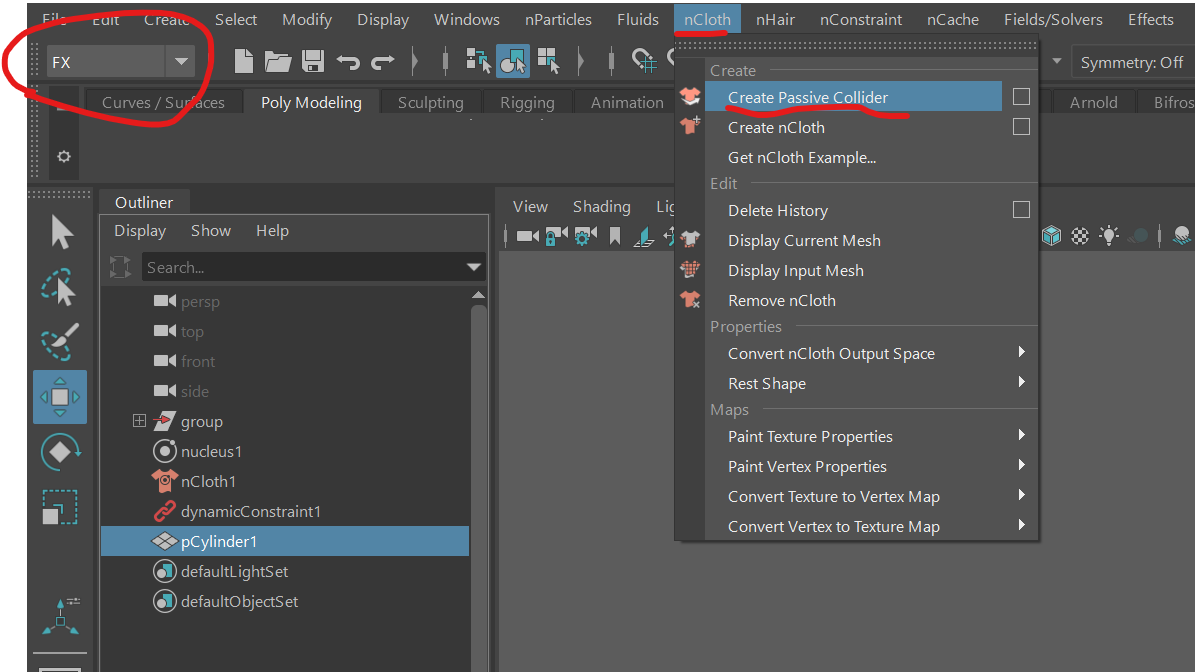


Rysunek 15 Wstawianie bryły

Korzystając ze strzałek do manipulacji wielkością bryły należy sprawić, by jej wysokość była zbliżona do wysokości modelu sukienki, natomiast obwód nie większy niż obwód modelu sukienki w najwęższym miejscu - chodzi o to, by bryły na siebie nie zachodziły, ale walec po przesunięciu znalazł się idealnie wewnątrz sukienki. Modelowanie wielkości bryły może się odbyć wewnątrz rozrywanego modelu albo obok, ważne, aby dobrze to porównać.

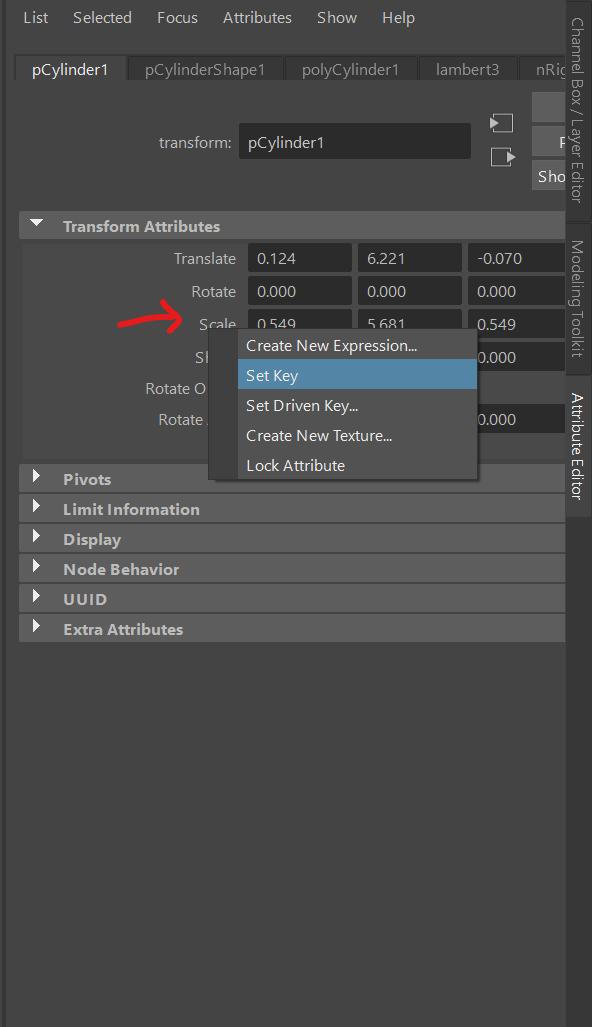
Następnie bryle należy nadać teksturę - najlepiej typu Lambert - której transparentność trzeba ustawić na sto procent. W finalnej animacji nie chcemy widzieć bryły walca, ale jedynie siłę, z jaką oddziałuje na model, który rozrywa.

Kiedy już tekstura bryły jest niewidoczna, a bryła umieszczona jest wewnątrz modelu, możemy wykonać ostatnie kroki w celu uzyskania animacji. Najpierw należy nadać bryle własności obiektu fizycznego (*Passive Collider*) z menu FX.

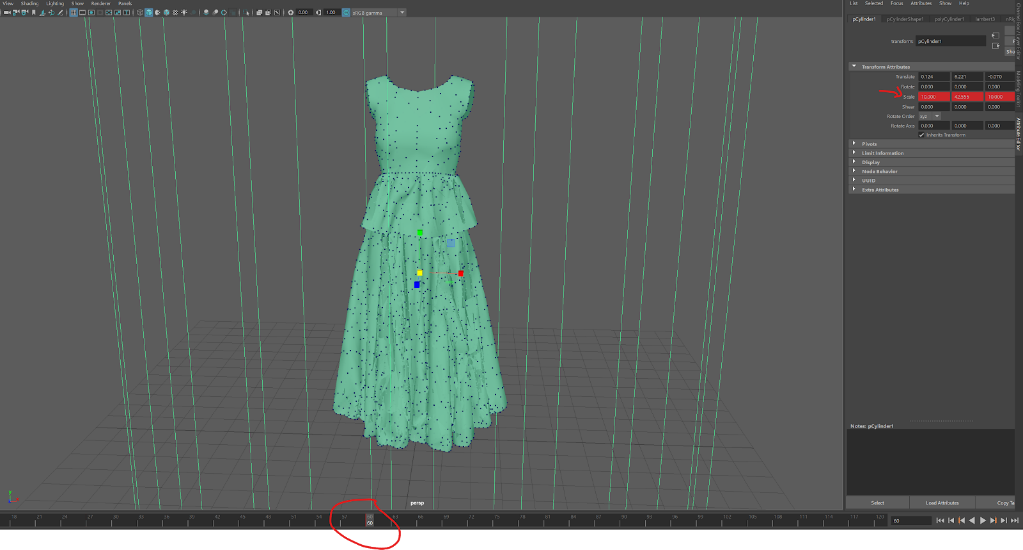


Rysunek 16 Utworzenie pasywnego obiektu

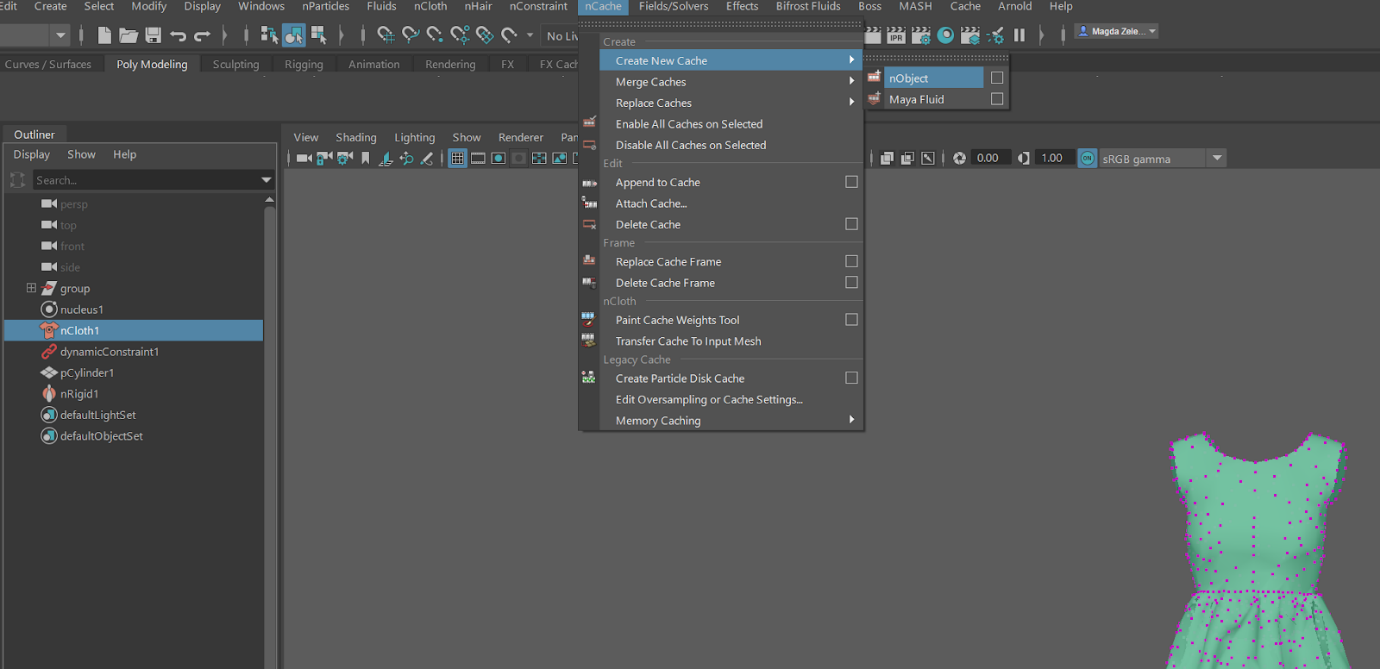
Następnie można przejść do samej animacji. Zatrzymując się na dowolnej klatce na osi czasu na dole okna programu należy zwrócić się do zakładki właściwości bryły walca. Jego obecną wielkość - mieszczącą się w obrębie sukienki - chcemy zapisać jako początkową wartości animacji, Klikając prawym przyciskiem myszy na Scale należy wybrać opcję Set Key (ustaw klucz animacji):



Klucz zostanie ustawiony na danej klatce animacji na danych wartościach. Obszar Scale powinien podświetlić się na czerwono.

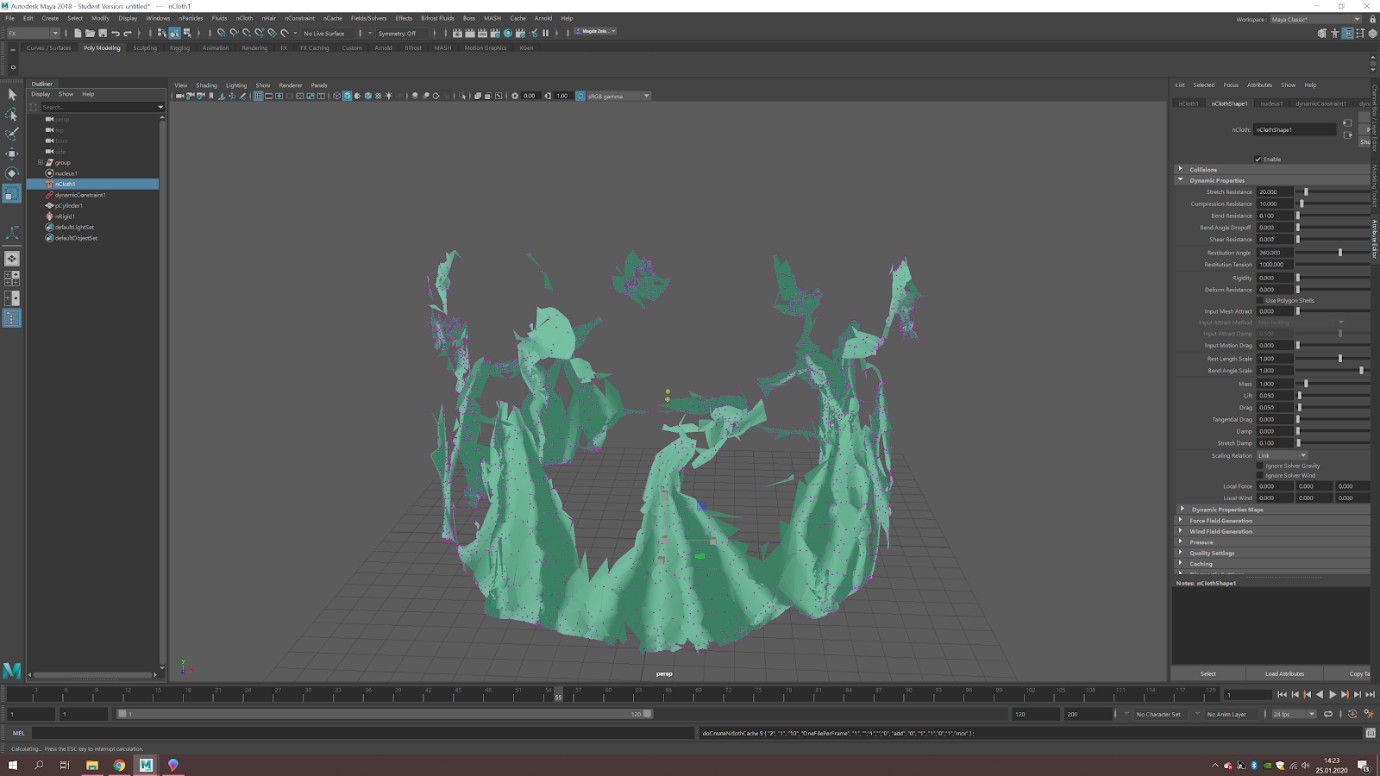
Następnie należy przesunąć się o kilkadziesiąt klatek dalej (tyle, ile chcemy, aby trwał nasz “wybuch”) i korzystając ze strzałek modyfikujących wielkość bryły lub edytując wartości x,y i z Scale zwiększyć walec tak, aby znacznie wychodził poza obszar sukienki. Następnie znów klikamy prawy przycisk na Scale i wybieramy Set Key.

Rysunek 17 Ustawianie kluczy animacji

W ten sposób zostało przygotowane już wszystko, co niezbędne do wywołania podglądu animacji. Można teraz kliknąć play przy osi czasu, ale znacznie szybszy i optymalniejszy podgląd animacji uzyska się, wykonując wstępne obliczenia i zapisując je w pamięci tymczasowej. W tym celu wykorzystamy narzędzie nCache. Należy zaznaczyć obiekt nCloth na liście obiektów, a następnie z menu górnego FX wybrać nCache -> Create new Cache -> nObject. W ten sposób zapamięta się wartość obiektu nCloth we wszystkich klatkach animacji.

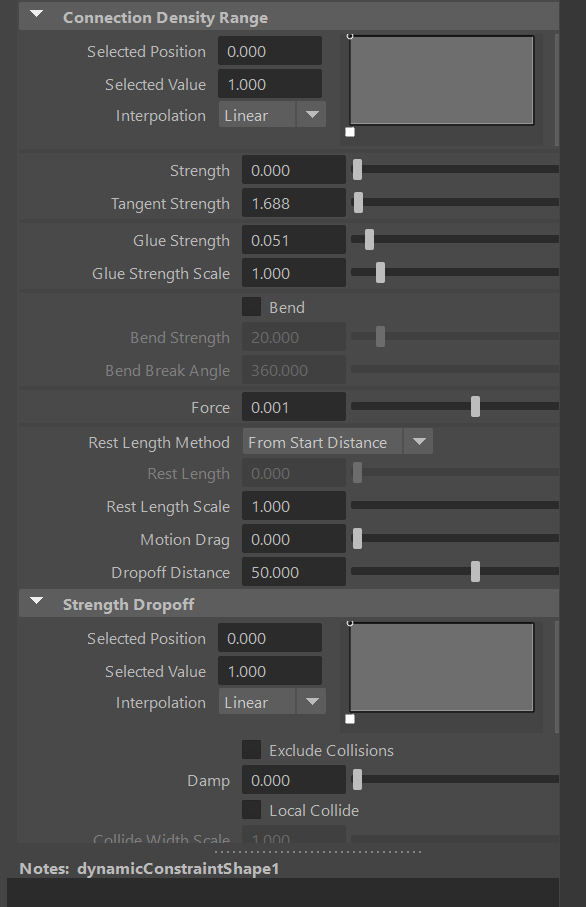
Rysunek 18 Pre-renderowanie animacji do pamięci tymczasowej cache

Teraz można chwilę poczekać na obliczenia. W skończonym czasie, dłuższym lub krótszym w zależności od skomplikowania modelu i wartości ustawień, nCache przeliczy całą animację i będzie można podejrzeć projekt w czasie rzeczywistym.

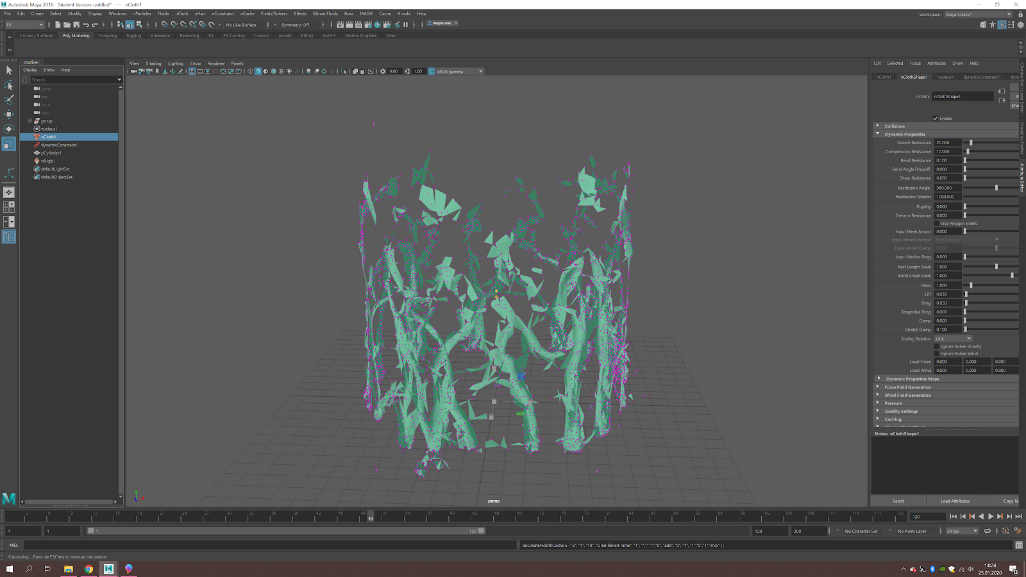


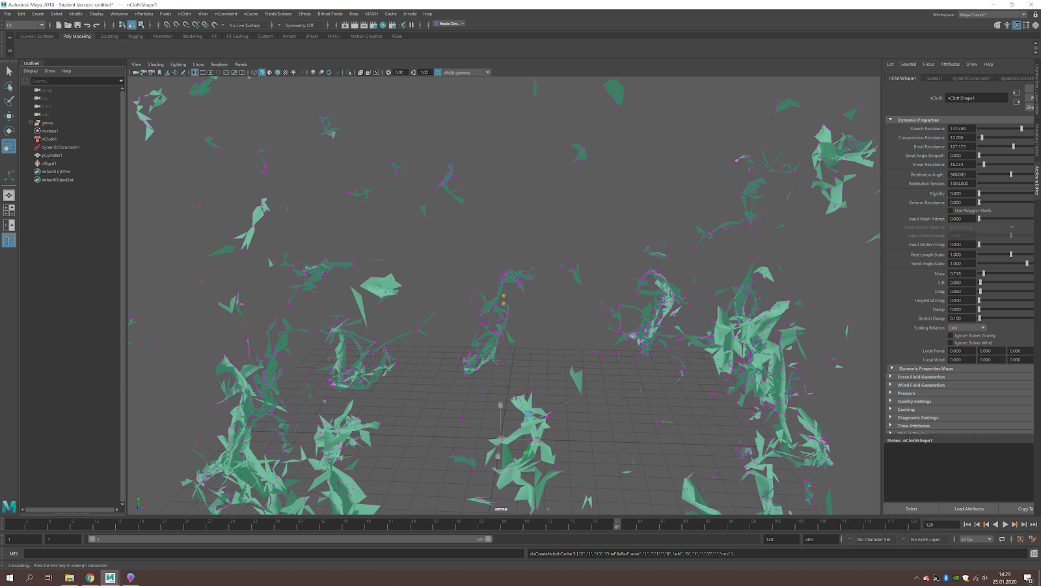
Rysunek 19 Efekt wybuchu modelu wg inicjalnych ustawień

Dzięki nCache możemy jeszcze w obrębie Maya zobaczyć dokładnie jaka jest dynamika i przebieg animacji oraz poprawić wartości przed renderingiem. Na tym przykładzie widać, że efekt nie został osiągnięty, materiał zostaje rozerwany na dosyć duże połacie, a chodziło o uzyskanie charakteru inspirowanego particles.

  
W celu poprawy można pobawić się ustawieniami Strength i Bend Resistance nCloth i sprawdzić, co pozwala uzyskać najlepsze efekty. W tym przykładzie jednak zwiększam nieco wartość siły rozpadu szwów w komponencie nConstraint (Tearable Surface). Ta właściwość ma dużą moc, więc zwiększenie jej o 0.001 jednostki powinno już znacząco wpłynąć na końcowy efekt.

Rysunek 20 Zmiana ustawień Tearable Surface

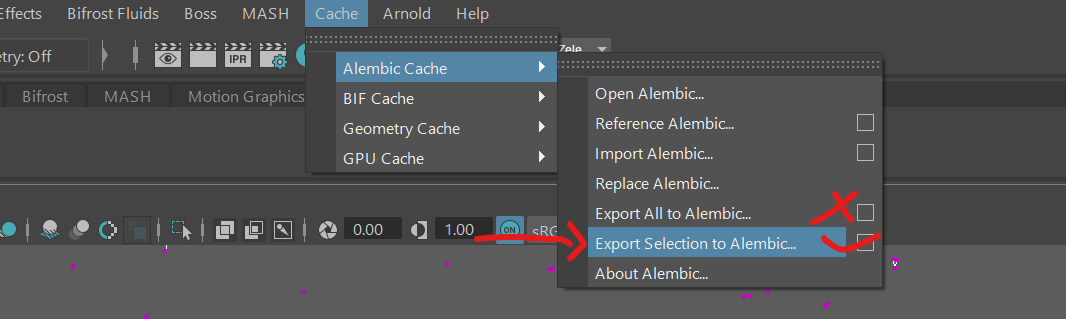




Rysunek 21 Efekt wybuchu po zmianie ustawień

Na powyższych obrazach widać, że jakość eksplozji znacznie się poprawiła, animacja również jest dynamiczniejsza i ma szerszy zakres, a elementy wirują szybciej. Dobierając odpowiednio wartości oraz kształt wewnętrznej bryły można uzyskać bardzo różne efekty w zależności od potrzeb.

W ten sposób uzyskałam animację eksplozji wszystkich modeli. Animacje wyeksportowałam z Maya w formacie .abc – Alembic, który jest najobszerniejszym rozszerzeniem do zapisu informacji o obiektach 3D. Animacje następnie połączyłam w całość w programie do edycji wideo Davinci Resolve, gdzie dodałam im również spójną barwę.



Rysunek 22 Eksport animacji do formatu Alembic

### Animacja ruchu tkaniny

Kolejnym przykładem użycia nCloth w tym projekcie jest symulacja zachowania powierzchni tkaniny podczas ruchu nadanego obiektowi. Proces dążący do osiągnięcia tego celu przebiegał następująco:

1. Tworzę obiekt typu walec, ustawiam go prostopadle do siatki sukienki, niewiele ponad nią
2. Nadaję walcu właściwości obiektu typu *Passive Collider*
3. Zaznaczam na sukience wierzchołki znajdujące się najbliżej walca, z wciśniętym klawiszem *shift*
4. Trzymając wciśnięty klawisz *shift* zaznaczam również siatkę walca
5. Z menu FX wybieram utworzenie więzów między obiektami
6. Animuję w czasie poziomy ruch walca od punktu początkowego w klatce 1 do oddalonego w dowolnej klatce (hipotetycznie 200)
7. Jeśli siatka sukienki nie ma nałożonego symulatora nCloth, tworzę symulator nCloth
8. Uruchamiam animację

Po poprawnym wykonaniu tych czynności obiekt sukienki podąży przyczepiony do walca, a struktura sukienki będzie falować niczym przymocowana do wieszaka.

1. Parisi, Tony, *Aplikacje 3D. Przewodnik po HTML5, WebGL i CSS3*, Helion 2015, str. 9 [↑](#footnote-ref-1)
2. MDN web docs, <https://developer.mozilla.org/pl/docs/Web/JavaScript> [↑](#footnote-ref-2)
3. Ibidem. [↑](#footnote-ref-3)
4. Parisi, Tony, *Aplikacje 3D. Przewodnik po HTML5, WebGL i CSS3*, Helion 2015, str. 22 [↑](#footnote-ref-4)
5. Andeggs - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=7504188 [↑](#footnote-ref-5)
6. Góra, Michał, Matematyka, E-podręcznik AGH, <https://epodreczniki.open.agh.edu.pl/tiki-index.php?page=P%C5%82aszczyzny+w+tr%C3%B3jwymiarowej+przestrzeni+rzeczywistej> [↑](#footnote-ref-6)
7. Ibidem. [↑](#footnote-ref-7)
8. Źródło i licencja: CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=337435 [↑](#footnote-ref-8)
9. Parafraza za: 1928 June 4, The Detroit Free Press, Paragraphs by Robert Quillen, Quote Page 6, Column 4, Detroit, Michigan. [↑](#footnote-ref-9)
10. https://www.theguardian.com/commentisfree/2019/jun/18/fast-fashion-environmental-audit-committee-polluting-industry [↑](#footnote-ref-10)
11. Dokumentacja Maya, https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2020/ENU/Maya-CharEffEnvBuild/files/GUID-ED791F1C-8412-4785-829F-9925F2604E8A-htm.html, tłumaczenie własne [↑](#footnote-ref-11)