

Proceduralne generowanie grafiki w grze Open Ski Jumping

(Procedural graphics generation
for video game Open Ski Jumping)

Jonatan Hrynkó

Praca licencjacka

Promotor: dr Łukasz Piwowar

Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Informatyki

1 września 2021



Streszczenie

Celem niniejszej pracy jest pokazanie jednego z możliwych sposobów na stworzenie proceduralnego generatora realistycznie wyglądających skoczni narciarskich. Rozwiązania te mogą znaleźć zastosowanie także w przypadku gier o innej tematyce niż skoki narciarskie. Obiekty proceduralne, których opis został przedstawiony w pracy pozwalają na stworzenie w oparciu o nie dodatkowych obiektów, które będą umożliwiały zwięzlejsze opisanie niektórych typów konstrukcji.

This paper aims to show one of the possible ways for creating a procedural ski jumping hills generator. These solutions can also be used in the case of video games with themes other than ski jumping. The procedural objects described in this paper might be a base for the creation of other objects, which will describe some types of constructions in a more concise way.

Spis treści

1. Wprowadzenie	7
2. Proceduralne generowanie geometrii	17
2.1. Proceduralne generowanie geometrii w silniku Unity	17
2.2. Krzywe	18
2.2.1. Line(A, B)	19
2.2.2. Arc(A, B)	19
2.2.3. Bezier(A, C1, C2, B)	19
2.3. Renderowanie ścieżek	22
2.4. Krzywe równoległe	22
2.5. Obiekty proceduralne	24
2.5.1. Construction	24
2.5.2. Stairs	27
3. Ukształtowanie terenu	31
3.1. Dane wysokościowe	31
3.2. Modyfikacja ukształtowania terenu	32
3.2.1. Inverse distance weighting	32
3.2.2. Modyfikacja metody inverse distance weighting	33
3.3. Dopasowanie ukształtowania terenu do skoczni	33
3.3.1. Pojedyncza skocznia	33
3.3.2. Kompleksy skoczni	35
4. Geometria skoczni narciarskiej	39

4.1.	Klasyfikacja skoczni narciarskich	39
4.2.	Parametry profilu skoczni narciarskiej	39
4.3.	Najazd	43
4.3.1.	Obliczenia	44
4.4.	Zeskok	47
4.4.1.	Bula i strefa lądowania	47
4.4.2.	Krzywa przejściowa	49
5.	Podsumowanie	53
	Bibliografia	57

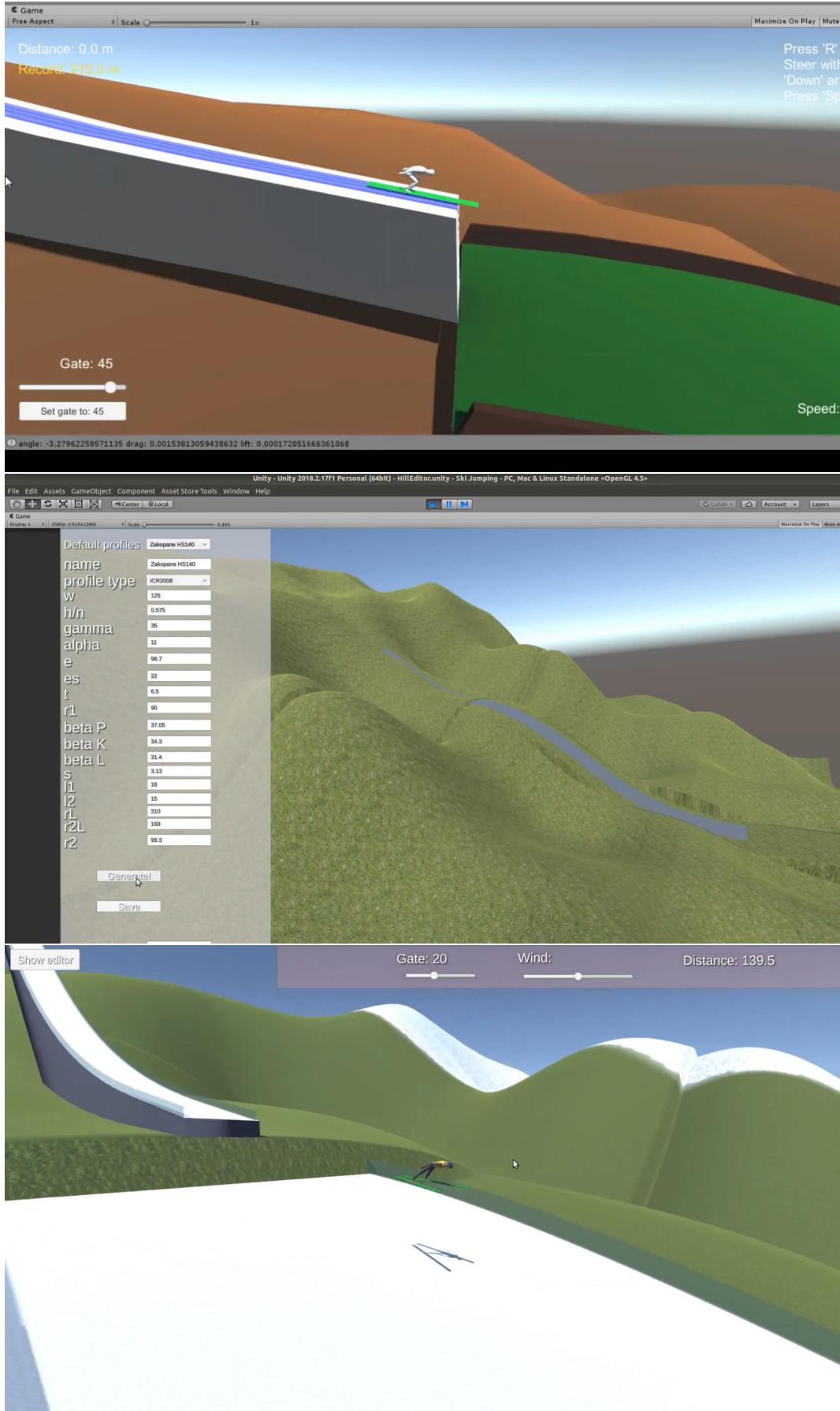
Rozdział 1.

Wprowadzenie

Gra Open Ski Jumping jest grą tworzoną przeze mnie od sierpnia 2018 roku. Kod źródłowy gry jest udostępniany na GitHubie [15] na licencji GNU GPL v3.0. Według danych GitHuba, gra ma już ponad 17000 pobrań.

Początkowo gra była bardzo uboga, gdyż umożliwiała oddanie skoku tylko z jednej skoczni. Ówczesny model gry wymagał stworzenia modelu skoczni w programie do modelowania 3D. Jednakże samo tworzenie modelu profilu skoczni było bardzo żmudnym procesem. Wymagało to przepisywania współrzędnych zapisanych na projektach skoczni, bądź też wygenerowanych za pomocą arkusza kalkulacyjnego, który obliczał współrzędne na podstawie certyfikatu skoczni.

Aby przyspieszyć ten proces, stworzyłem narzędzie, które umożliwia wygenerowanie modelu profilu skoczni. Pierwotna wersja tego narzędzia była wtyczką do programu Blender, finalna projektem w Unity. Generator skoczni używa danych z certyfikatów FIS (Międzynarodowa Federacja Narciarska, skrót od wersji francuskiej: Fédération Internationale de Ski), co umożliwiło odtworzenie wirtualnych modeli wszystkich skoczni znanych z zawodów międzynarodowych w bardzo krótkim czasie. Poza możliwością generowania istniejących skoczni, generator pozwala graczom na tworzenie własnych.



Rysunek 1.1: Początkowe wersje gry

Kolejne etapy projektu, poza poprawą wyglądu samej skoczni skupiały się na systemie rozgrywania konkursów. Początkowa wersja gry nie pozwalała na ich rozgrywanie, późniejsza wersja umożliwiała rozegranie konkursu z dwoma seriami. Docelowym celem było odtworzenie wszystkich konkursów i turniejów, które są rozgrywane podczas Pucharu Świata w skokach narciarskich. Do tej pory, gry o tematyce skoków narciarskich umożliwiały stworzenie tylko jednej klasyfikacji punktowej, co nie oddawało realiów prawdziwych zawodów. Zabrakło możliwości łączenia konkursów drużynowych i indywidualnych w jednym turnieju.

Konkursy Pucharu Świata w skokach narciarskich w większości przypadków są zaliczane tylko do klasyfikacji generalnej Pucharu Świata oraz klasyfikacji generalnej Pucharu Narodów (klasyfikacja drużynowa - punkty danego kraju to suma punktów z Pucharu Świata zawodników reprezentujących dany kraj). Zdecydowana większość konkursów poprzedzona jest konkursem kwalifikacyjnym, w którym wyłaniana jest odpowiednia liczba skoczków (50 lub 40), którzy będą mogli przystąpić do konkursu głównego. Konkurs główny składa się z dwóch serii, gdzie w serii finałowej prawo startu przysługuje 30 najlepszym zawodnikom pierwszej serii.

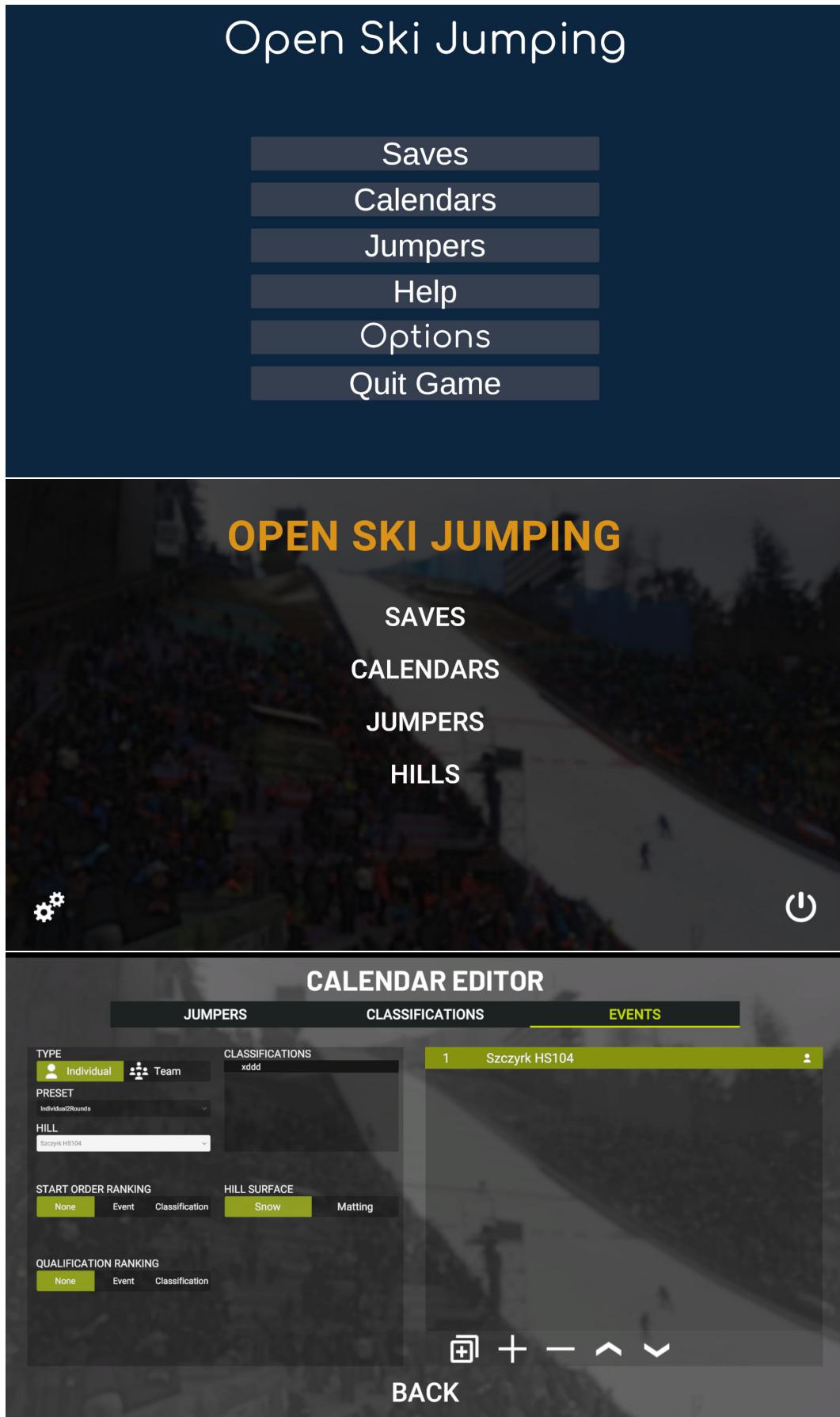
Jednakże oprócz takich *standardowych* konkursów, podczas sezonu rozgrywa się także konkursy, które są zaliczane również do innych klasyfikacji, takich jak Puchar Świata w Lotach, Turniej Czterech Skoczni, Raw Air, czy też tak zwanych miniturniejów, takich jak Planica 7, Willingen 5 lub Titisee-Neustadt 5. O ile Turniej Czterech Skoczni, pomimo nieco innego formatu rozgrywania konkursów, ma dosyć klasyczną formułę - do klasyfikacji turnieju zaliczane są noty punktowe z poszczególnych skoków oddanych w konkursach, to w innych turniejach do punktacji zalicza się także wyniki z kwalifikacji. W przypadku Raw Air i Planica 7 zalicza się również punkty uzyskane przez poszczególnych zawodników w konkursach drużynowych. W przypadku turnieju Willingen 5, rozgrywa się jeden konkurs kwalifikacyjny do dwóch konkursów, co daje 5 skoków, które są zaliczane do klasyfikacji końcowej. Innym konkursem odbiegającym od normy swoją formułą jest ostatni konkurs Pucharu Świata - nie jest rozgrywany do niego konkurs kwalifikacyjny, gdyż prawo startu w nim ma tylko czołowa trzydziestka klasyfikacji generalnej Pucharu Świata.

Tworząc system rozgrywania konkursów, który mógłby obsłużyć te wszystkie niestandardowe konkursy, można to zrobić na przynajmniej dwa sposoby. Jedną z możliwości byłoby obsłużenie wszystkich możliwych przypadków niestandardowych konkursów i turniejów. Niestety każda zmiana w zasadach powodowałaby konieczność modyfikacji systemu konkursów. Innym możliwym rozwiązaniem byłoby stworzenie bardziej ogólnego systemu, za pomocą którego można by opisać wszystkie potrzebne konkursy i turnieje (i ta opcja została wybrana).

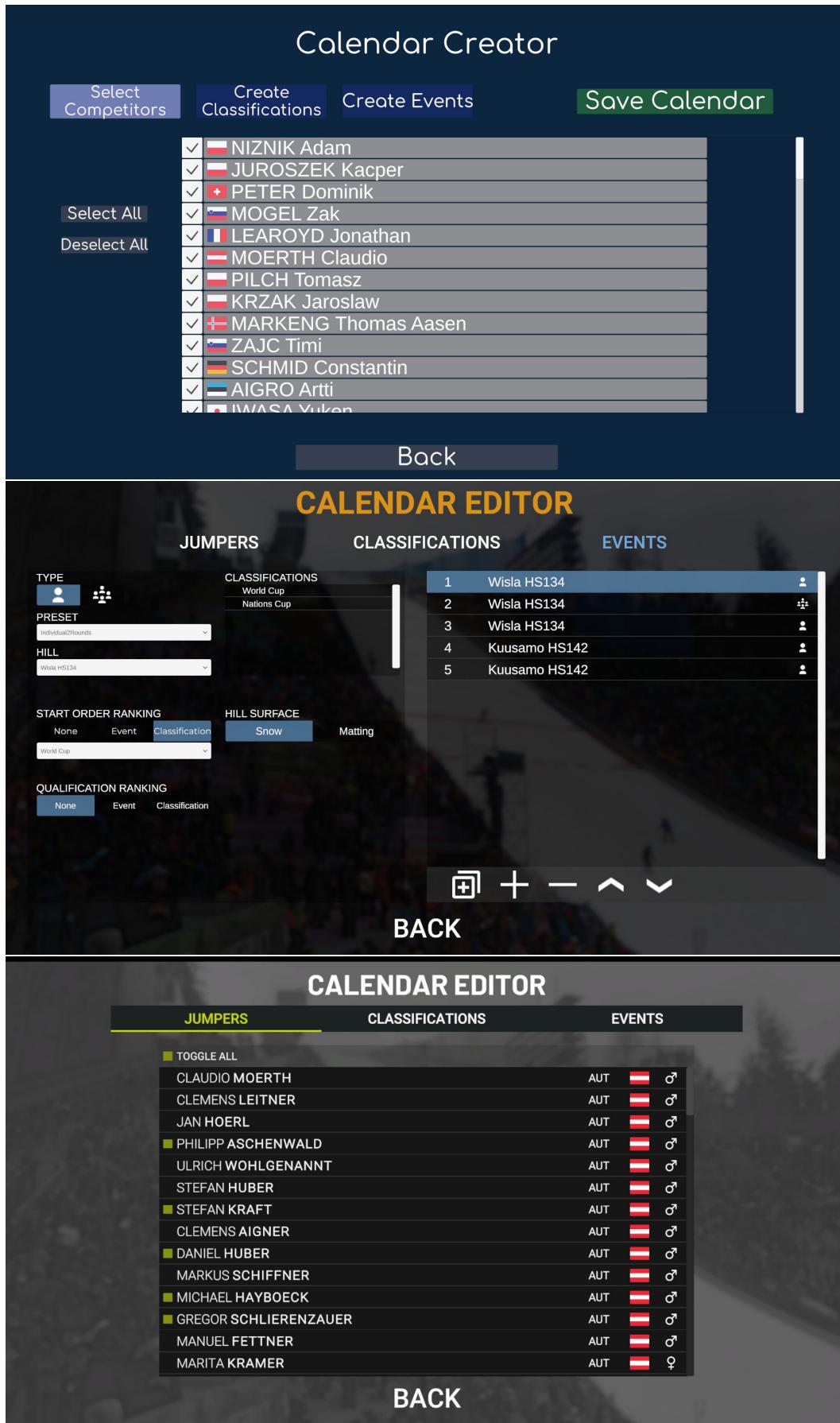
W pierwszej aktualizacji systemu konkursów dodana została możliwość rozgrywania konkursów indywidualnych wraz z możliwością ich dowolnej konfiguracji. Możliwe było ustalenie konkursu kwalifikacyjnego dla danego konkursu, wybór klasyfikacji, do których wyniki tego konkursu będą zaliczane, czy też stworzenie

własnego formatu konkursu (na przykład konkursu składającego się z trzech serii, bądź takiego gdzie zamiast 30 zawodników, do kolejnej serii awansuje 20). W drugiej aktualizacji, system konkursów został wzbogacony o możliwość łączenia konkursów indywidualnych z drużynowymi w jednym sezonie. Dzięki temu system konkursów pozwalał na opisanie wszystkich turniejów i konkursów występujących w Pucharze Świata.

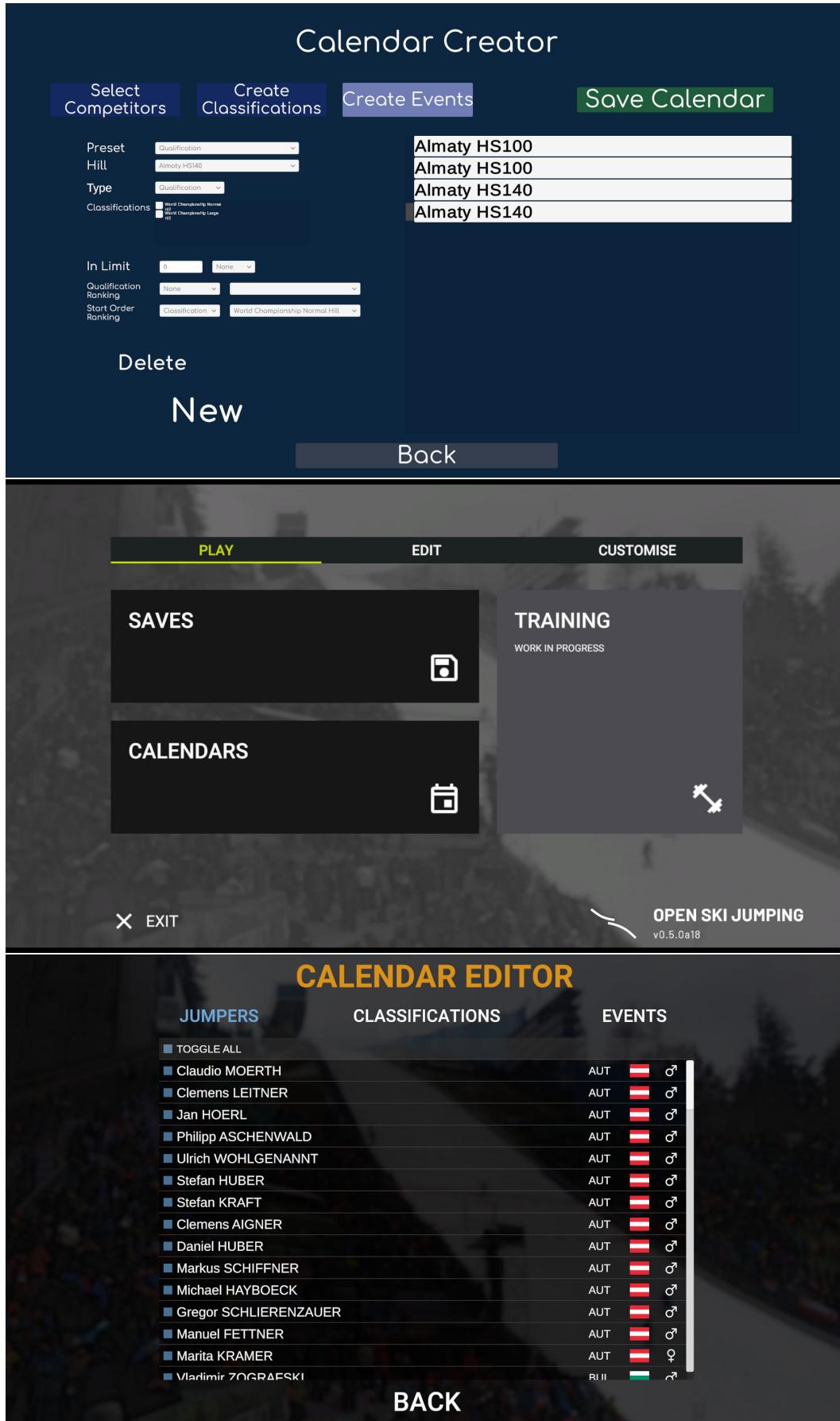
Oprócz udoskonalenia systemu konkursów, stopniowo poprawiany był interfejs graficzny. Ponadto sam kod gry i zależności pomiędzy klasami, również uległy zmianom.



Rysunek 1.2: Porównanie wyglądu poszczególnych elementów menu głównego w różnych wersjach gry

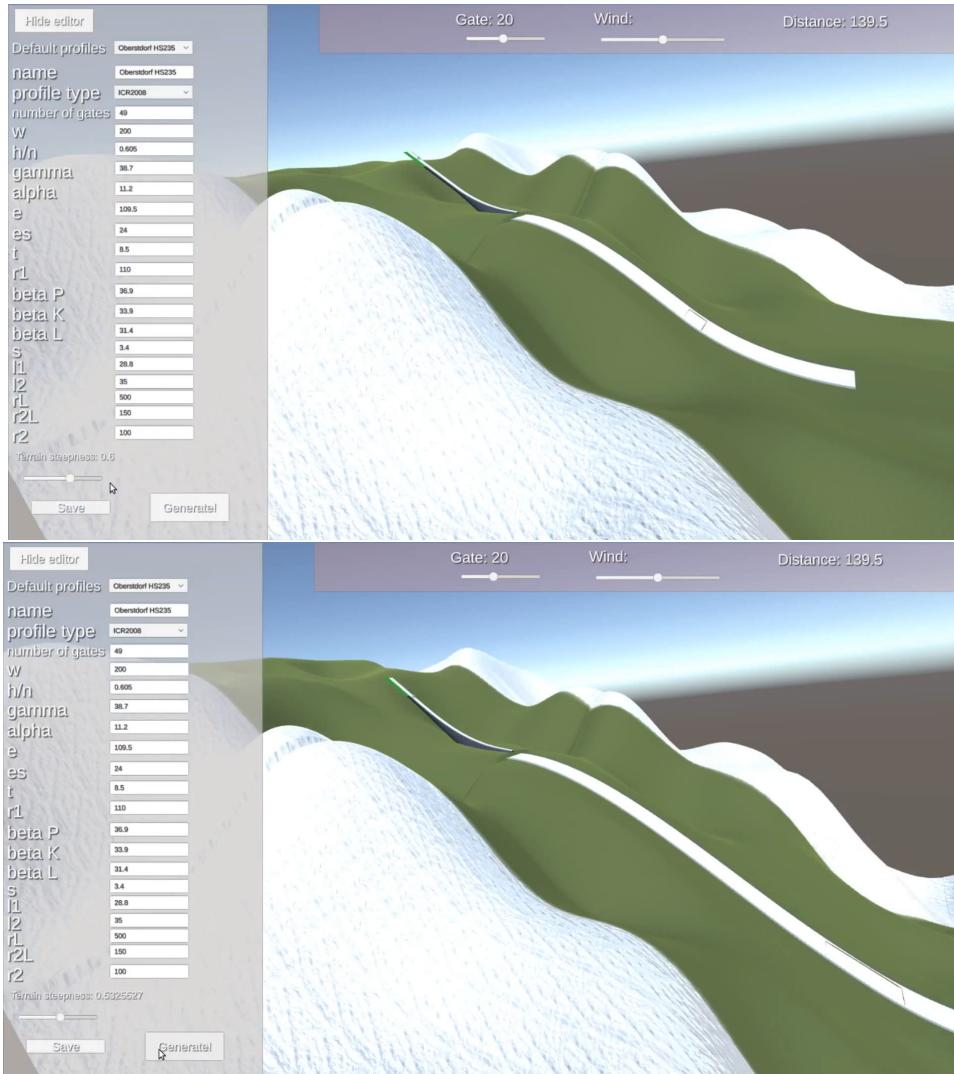


Rysunek 1.3: Porównanie wyglądu poszczególnych elementów menu głównego w różnych wersjach gry



Rysunek 1.4: Porównanie wyglądu poszczególnych elementów menu głównego w różnych wersjach gry

Po poprawieniu systemu konkursów, potrzebne było dalsze udoskonalanie generatora skoczni. Głównym jego problemem było to, że wszystkie skocznie wyglądały niemal identycznie - pozwalał on jedynie na ustawienie ukształtowania terenu wokół rozbiegu za pomocą parametru **terrainSteepness**.



Rysunek 1.5: Ukształtowanie terenu w zależności od wartości **terrainSteepness**

Pierwszą modyfikacją generatora, oprócz ogólnej zmiany w generowaniu konstrukcji skoczni, było zastosowanie kilku dodatkowych parametrów. Były one odpowiedzialne m.in. za to, po której stronie skoczni znajdowały się schodki prowadzące na belkę startową, czy też umożliwiała zmianę rodzaju torów najazdowych. W późniejszym czasie, pojawiły się także pierwsze prototypy kompleksów skoczni, jednakże problemem było generowanie terenu wokół skoczni w taki sposób, by nie pojawiały się artefakty.



Rysunek 1.6: Działanie generatora skoczni po wzbogaceniu o większą liczbę parametrów

16:73044

Rozdział 2.

Proceduralne generowanie geometrii

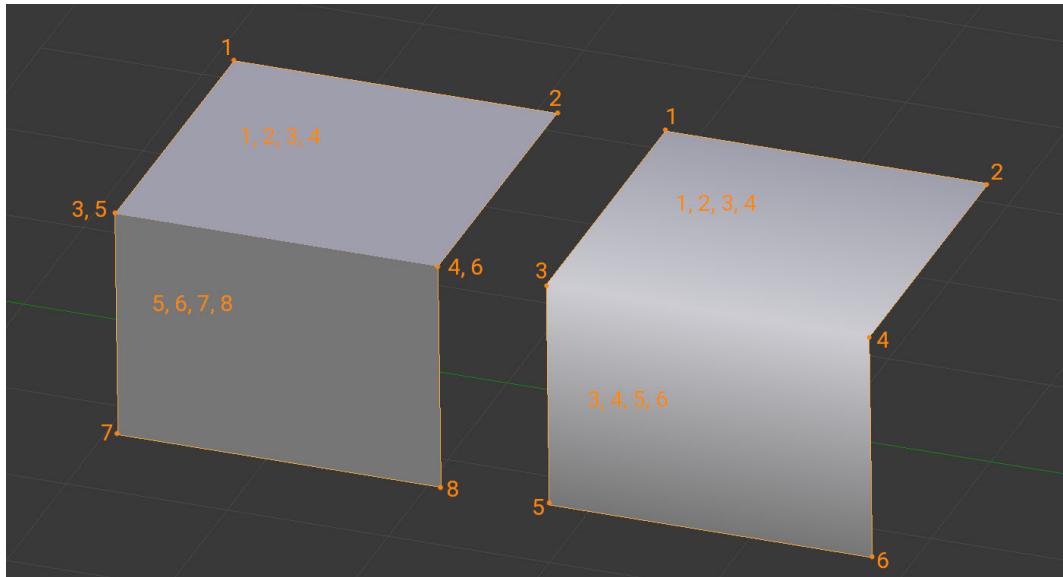
2.1. Proceduralne generowanie geometrii w silniku Unity

Silnik Unity reprezentuje geometrię modelu trójwymiarowego za pomocą klasy **Mesh** [7]. Klasa **Mesh** utrzymuje m.in.: wierzchołki(`Vector3[]`), indeksy wierzchołków trójkątów(`int[]`), współrzędne tekstur - `uv1`, `uv2`, ..., `uv8` (`Vector2[]`), kolory wierzchołków(`Color[]`), jak również wektory normalne wierzchołków (`Vector3[]`). Wszystkie te dane mogą być wykorzystywane w shaderach. Jako, że dla jednego obiektu typu **Mesh** można ustawić tylko jeden materiał, silnik Unity pozwala na podzielenie jednego obiektu na podobiekty, określone jako **Submesh**.

Domyślnym sposobem reprezentacji geometrii trójwymiarowej w Unity jest siatka trójkątów, która jest reprezentowana za pomocą listy wierzchołków i listy indeksów wierzchołków trójkątów. Aby stworzyć model trójwymiarowy z poziomu kodu należący do danego obiektu, obiekt ten musi posiadać komponenty **MeshFilter** [11] oraz **MeshRenderer** [12]. W przypadku komponentu **MeshFilter** należy przypisać jemu informacje o geometrii modelu zawarte w instancji klasy **Mesh**, natomiast w komponencie **MeshRenderer** potrzebne będzie ustawienie materiału, który będzie użyty do renderowania tego modelu.

Aby opisać geometrię modelu za pomocą klasy **Mesh**, należy ustawić wierzchołki, trójkąty, a także uv (potrzebne, aby materiał się prawidłowo renderował). Trzeba jednak pamiętać o tym, że ustawienie właściwości `triangles` powoduje sprawdzenie poprawności indeksów, także w przypadku gdy nie ustawimy wierzchołków, wszystkie indeksy będą poza zakresem. Dlatego ustawianie właściwości `Mesh`, należy rozpocząć od ustawienia wierzchołków, następnie można ustawić współrzędne uv wierzchołków, bądź też indeksy trójkątów. Po ustawieniu wierzchołków oraz trójkątów, możemy wywołać funkcję, która dokona obliczenia wektorów normalnych w wierzchołkach. W tym miejscu należy zaznaczyć, że funkcja `RecalculateNormals` za-

kłada, iż wszystkie krawędzie, które łączą dwie ściany mają być cieniowane w sposób gładki. Aby zatem uzyskać ostre krawędzie, należy utworzyć kopię wierzchołków leżących na krawędziach łączących ściany, tak jak to zostało pokazane na Rysunku 2.1.



Rysunek 2.1: Uzyskiwanie ostrych krawędzi za pomocą podwojenia wierzchołków

Mając gotowe informacje o geometrii modelu zawarte w instancji klasy Mesh, możemy przypisać je do odpowiedniego atrybutu MeshFilter. Na koniec pozostało przypisać odpowiedni materiał do MeshRenderer. Poniżej został zamieszczony krótki kod ilustrujący powyższy opis.

```
var meshFilter = gameObject.GetComponent<MeshFilter>();
var meshRenderer = gameObject.GetComponent<MeshRenderer>();

var mesh = new Mesh();
mesh.vertices = vertices;
mesh.triangles = triangles;
mesh.uv = uvs;
mesh.RecalculateNormals();

meshFilter.mesh = mesh;
meshRenderer.material = material;
```

2.2. Krzywe

Jednym z podstawowych założeń generatora skoczni jest możliwość zmian w profilu skoczni, bez zepsucia już istniejącej konstrukcji. Konieczne zatem jest stwo-

rzenie pewnych narzędzi umożliwiających opisanie różnych elementów konstrukcyjnych skoczni narciarskiej, jak i samego profilu skoczni. W związku z tym jednym ze składników generatora są ścieżki mogące składać się z różnego rodzaju krzywych. Wszystkie krzywe opisane poniżej będą opisywane parametrycznie, przy czym parametryzacja będzie proporcjonalna względem długości krzywej.

2.2.1. Line(A, B)

Podstawową krzywą, którą zawiera generator jest oczywiście odcinek - **Line**. Odcinek $S = \overrightarrow{AB}$ jest kombinacją wypukłą punktów **A** oraz **B**, zatem możemy go wyrazić parametrycznie jako $S(t) = (1 - t)\mathbf{A} + t\mathbf{B}$. Taka parametryzacja odcinka jest proporcjonalna do jego długości.

2.2.2. Arc(A, B)

Pomiędzy punktami **A**, **B** w przestrzeni istnieje nieskończenie wiele łuków. Należy zatem wprowadzić dodatkowe parametry, dzięki którym będzie można ograniczyć liczbę łuków przechodzących przez te punkty.

Jednym ze sposobów może być wprowadzenie dodatkowego punktu, przez który ma przechodzić łuk. Ponieważ przez te trzy punkty ma przechodzić łuk, punkty te nie mogą być współliniowe, zatem jednoznacznie wyznaczają płaszczyznę na której leży okrąg, a za pomocą trzech punktów możemy go wyznaczyć. Wówczas środek okręgu na którym leży ten wyznaczamy korzystając z iloczynu wektorowego, a następnie go parametryzujemy korzystając z interpolacji sferycznej (funkcja **Unity-Engine.Mathf.Slerp**).

Innym sposobem, na ustalenie przebiegu łuku może być dodatkowe określenie środka okręgu na którym leży, a także bitu mówiącego, czy mamy brać pod uwagę łuk o kącie mniejszym, czy też większym niż 180 stopni. Wówczas, podobnie jak w poprzednim przypadku możemy skorzystać z interpolacji sferycznej.

2.2.3. Bezier(A, C1, C2, B)

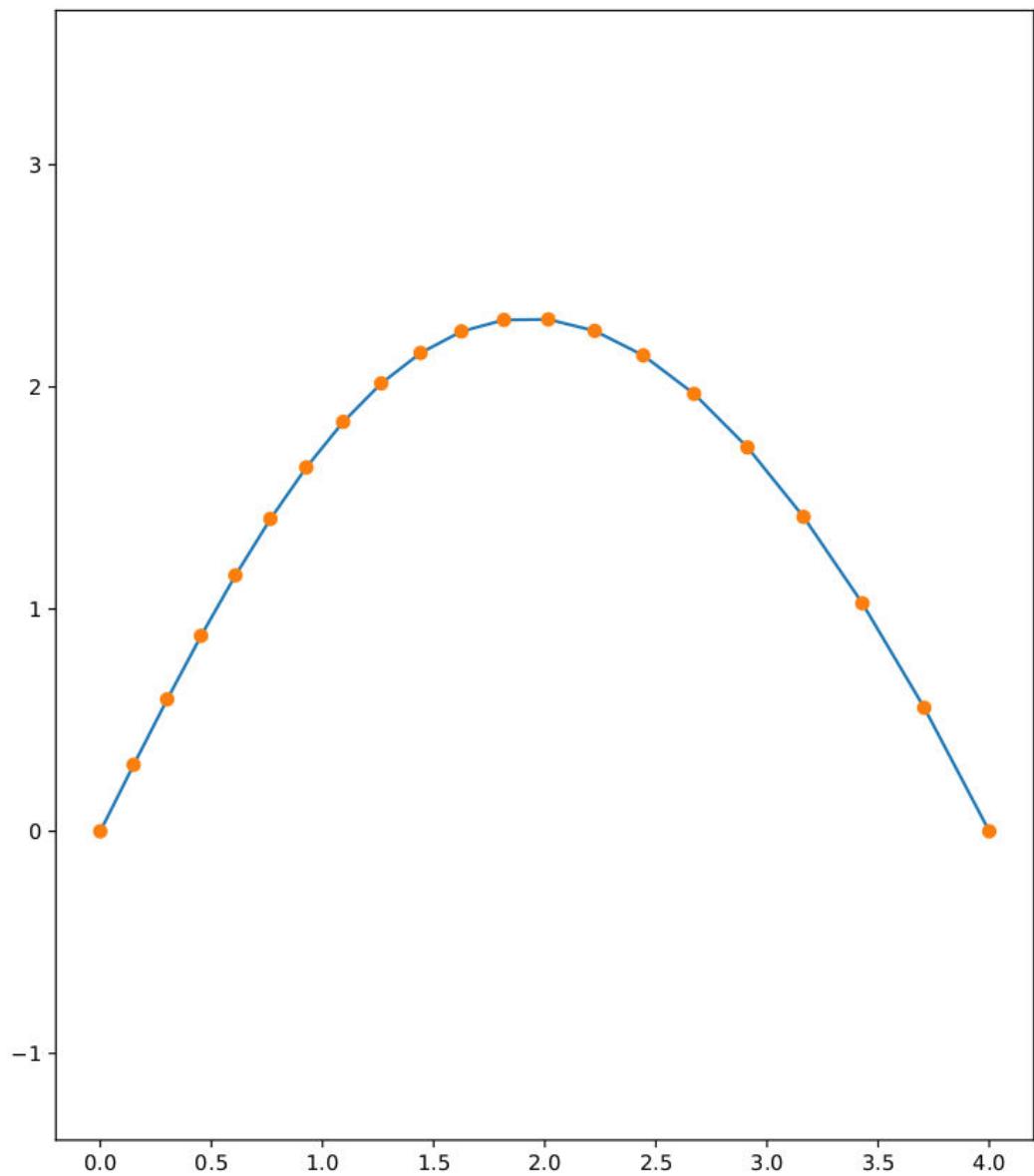
Kolejnym rodzajem krzywej, która jest wykorzystana w generatorze jest krzywa Beziera stopnia trzeciego. Wyznaczenie dowolnej parametryzacji krzywej Beziera nie jest zadaniem trudnym - wystarczy skorzystać z algorytmu de Casteljau.

$$\text{Lerp}(\mathbf{A}, \mathbf{B}, t) = (1 - t)\mathbf{A} + t\mathbf{B}$$

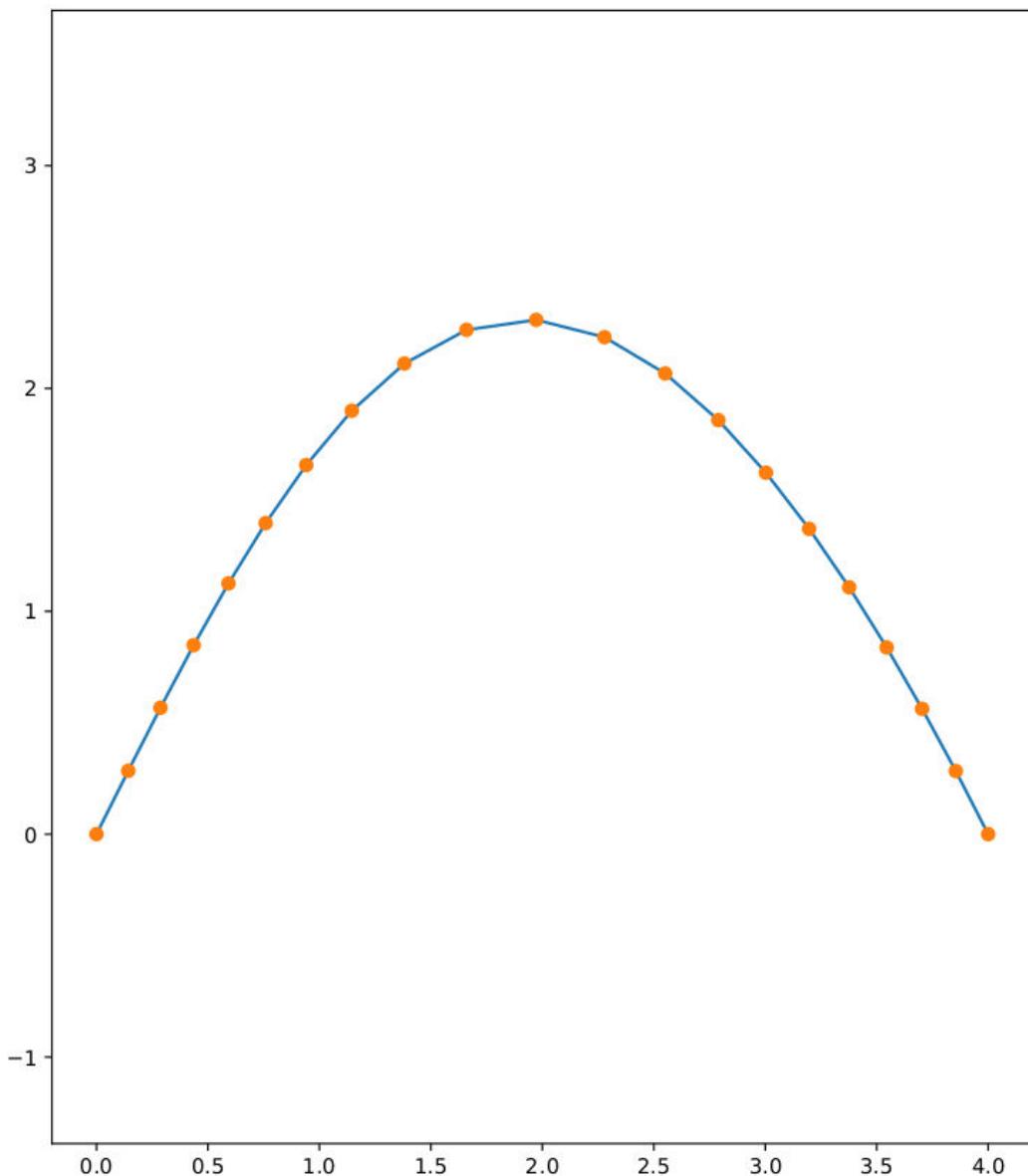
$$\text{Bezier2}(\mathbf{A}, \mathbf{B}, \mathbf{C}, t) = \text{Lerp}(\text{Lerp}(\mathbf{A}, \mathbf{B}, t), \text{Lerp}(\mathbf{B}, \mathbf{C}, t), t)$$

$$\text{Bezier3}(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, t) = \text{Lerp}(\text{Bezier2}(\mathbf{A}, \mathbf{B}, \mathbf{C}, t), \text{Bezier2}(\mathbf{B}, \mathbf{C}, \mathbf{D}, t), t)$$

Znacznie ciekawszym zagadnieniem jest wyznaczenie takiej parametryzacji, która będzie proporcjonalna względem długości krzywej.



Rysunek 2.2: W przypadku ewaluacji krzywej Beziera ze stałym przyrostem t , odległości pomiędzy kolejnymi punktami nie są równe.



Rysunek 2.3: W przypadku ewaluacji krzywej Beziera z przyrostem t proporcjonalnym do odwrotności jej znormalizowanej funkcji długości, odległości pomiędzy kolejnymi punktami są równe.

Aby uzyskać taką parametryzację krzywej Beziera, najpierw należy wyznaczyć najpierw funkcję jej długości $\bar{l}(t)$. Dzieląc wartości funkcji $\bar{l}(t)$, przez długość całej krzywej, czyli $\bar{l}(1)$, otrzymamy funkcję o wartościach z przedziału $[0, 1]$. Funkcję tą dalej będziemy nazywać **znormalizowaną** funkcją długości krzywej, a wyrażać się będzie ona wzorem:

$$l(t) = \frac{l(t)}{\bar{l}(1)}$$

Ponieważ znormalizowana funkcja długości krzywej jest ścisłe rosnąca, to istnieje jej

funkcja odwrotna $l^{-1}(t)$. Podstawiając $l^{-1}(t)$ w miejsce t , a następnie wstawiając to do wzoru krzywej Beziera otrzymamy:

$$B_{eq}(t) = B(l^{-1}(t))$$

W przypadku krzywej B_{eq} , jej długość jest proporcjonalna do jej parametryzacji.

Obliczenie przybliżenia $l(t)$, nawet za pomocą tak prostej metody jak aproksymacja krzywej łamaną, daje bardzo dobre efekty. Metoda ta w przypadku silnika Unity jest szczególnie prosta, gdyż możemy wykorzystać **AnimationCurve** [9] do reprezentacji funkcji $l^{-1}(t)$. Dzięki temu nie trzeba samodzielnie implementować ewaluacji funkcji w postaci punktowej.

2.3. Renderowanie ścieżek

Pomimo tego, że krzywe opisują geometrię w sposób bardzo zwięzły i łatwy w modyfikacji, geometria w Unity musi być opisana w postaci siatki trójkątów. Oznacza to, że krzywe muszą zostać przekształcone na łamane, które będą je aproksymowały. W przypadku generowania skoczni narciarskich, dosyć dobre efekty daje aproksymacja o stałą odległość na krzywej, na przykład co 1 metr. W momencie, gdy potrzebna jest większa precyzja, można ustawić tą wartość indywidualnie dla każdej ścieżki. Próbkowanie ścieżek ze stałym odstępem przydatne jest również w przypadku generowania schodów, gdyż wtedy uzyskujemy równe długości stopni.

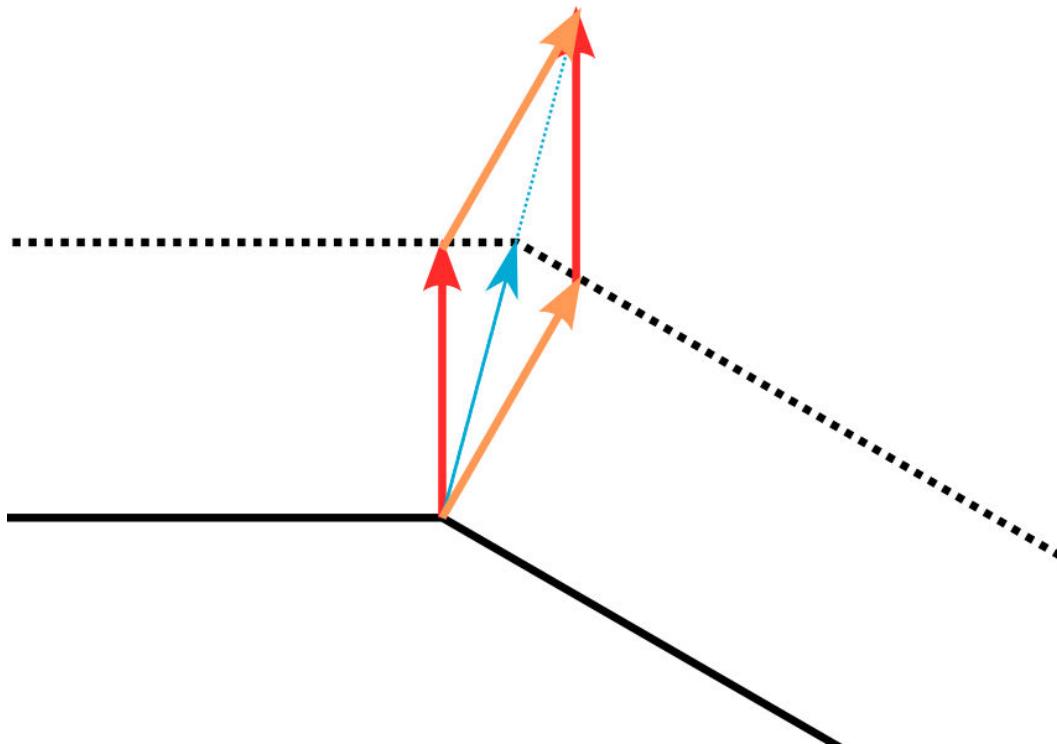
2.4. Krzywe równoległe

W przypadku generowania geometrii zarówno dwuwymiarowej jak i trójwymiarowej wzdłuż określonej krzywej, stosunkowo często potrzebne jest wyznaczenie krzywej równoległej. W przypadku proceduralnego generowania skoczni, wyznaczenie krzywej równoległej będzie przydatne przy generowaniu band, barierek, schodów, czy innych elementów konstrukcyjnych.

W przypadku dwuwymiarowym, aby uzyskać łamankę równoległą do innej łamanej najpierw należy wyznaczyć wektory normalne wszystkich segmentów tej łamanej. Możemy je obliczyć wykorzystując następujący fakt:

$$\begin{pmatrix} a \\ b \end{pmatrix} \perp \begin{pmatrix} -b \\ a \end{pmatrix}$$

Po obliczeniu kierunku wektora normalnego, musimy go jeszcze znormalizować, gdyż tak uzyskany wektor prostopadły będzie miał taką długość równą długości segmentu. Kolejnym krokiem będzie obliczenie wektora odsunięcia dla każdego wierzchołka wewnętrznego łamanej. Aby punkty przesunięte o ten wektor były



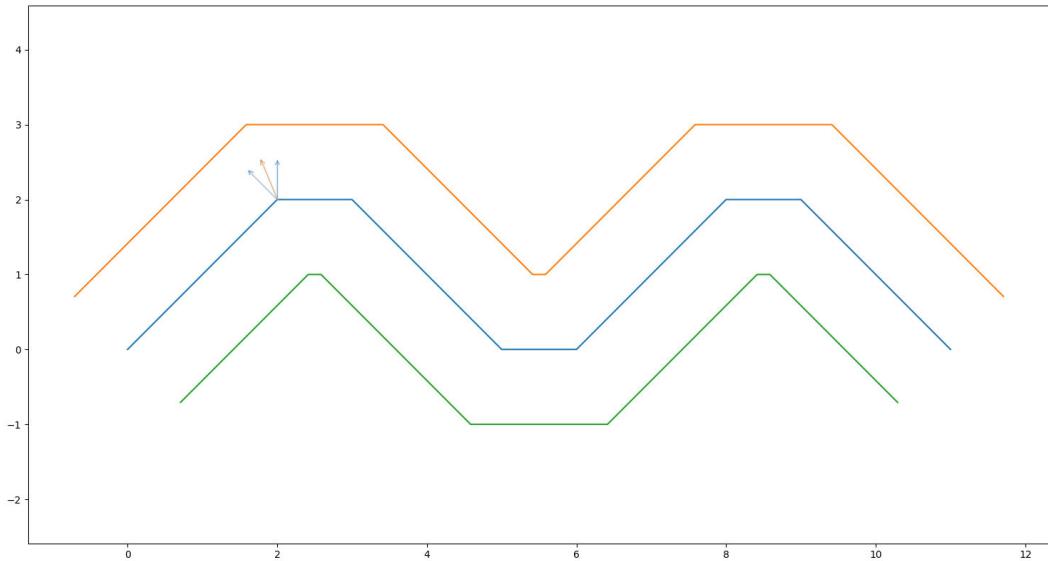
Rysunek 2.4: Zilustrowanie sposobu obliczania wektora odsunięcia

równo oddalone od obu z odcinków, wektor ten musi być równoległy do dwusiecznej kąta pomiędzy odcinkami. Wektory normalne odcinków oznaczymy jako n_i dla $i = 0, 1, \dots, n-2$. Wówczas wektor równoległy do wektora odsunięcia wierzchołka pomiędzy segmentami i -tym oraz $i+1$ -szym, to $v'_i = n_i + n_{i+1}$, dla dla $i = 0, 1, \dots, n-2$. Następnie potrzebne będzie obliczenie właściwego wektora odsunięcia. Zauważmy, że rzut tego wektora na wektor normalny sąsiadniego odcinka jest równy temu wektorowi normalnemu. Otrzymujemy zatem:

$$\begin{aligned} v_i &= \alpha v'_i \\ n_i \frac{v_i \cdot n_i}{n_i \cdot n_i} &= n_i \frac{\alpha v'_i \cdot n_i}{n_i \cdot n_i} = \alpha n_i \frac{v'_i \cdot n_i}{n_i \cdot n_i} = n_i \\ \alpha &= \frac{n_i \cdot n_i}{v'_i \cdot n_i} \end{aligned}$$

Otrzymany wektor v_i jest wektorem przesunięcia dla wierzchołka pomiędzy i -tym, a $i+1$ -szym segmentem łamanej. Wektorami przesunięcia dla dwóch skrajnych wierzchołków łamanej, będą wektory normalne sąsiadujących z nimi odcinków łamanej.

W przypadku trójwymiarowym zasada działania algorytmu jest podobna, jednakże trudność sprawia fakt, że zamiast prostej prostopadłej do odcinka, otrzymujemy płaszczyznę prostopadłą. Ponadto trzeba wyznaczyć dwa wektory prostopadłe



Rysunek 2.5: Wynik działania algorytmu dla przykładowej łamanej

do odcinków, jak i dwa wektory przesunięć w wierzchołkach: pierwszego, skierowanego w prawą stronę krzywej oraz drugiego, skierowanego w góre krzywej. W tym przypadku założymy również, że krzywa nie może zawierać fragmentów skierowanych pionowo. Wówczas wektor prostopadły skierowany w bok uzyskamy obliczając iloczyn wektorowy wektora kierunkowego danego odcinka z wektorem skierowanym w górę. Następnie, aby obliczyć wektor normalny odcinka skierowany w górę, obliczamy iloczyn wektorowy wektora kierunkowego odcinka z wektorem otrzymanym poprzednio. Aby obliczyć wektory przesunięć, stosujemy algorytm dla przypadku dwuwymiarowego dla obu wektorów normalnych.

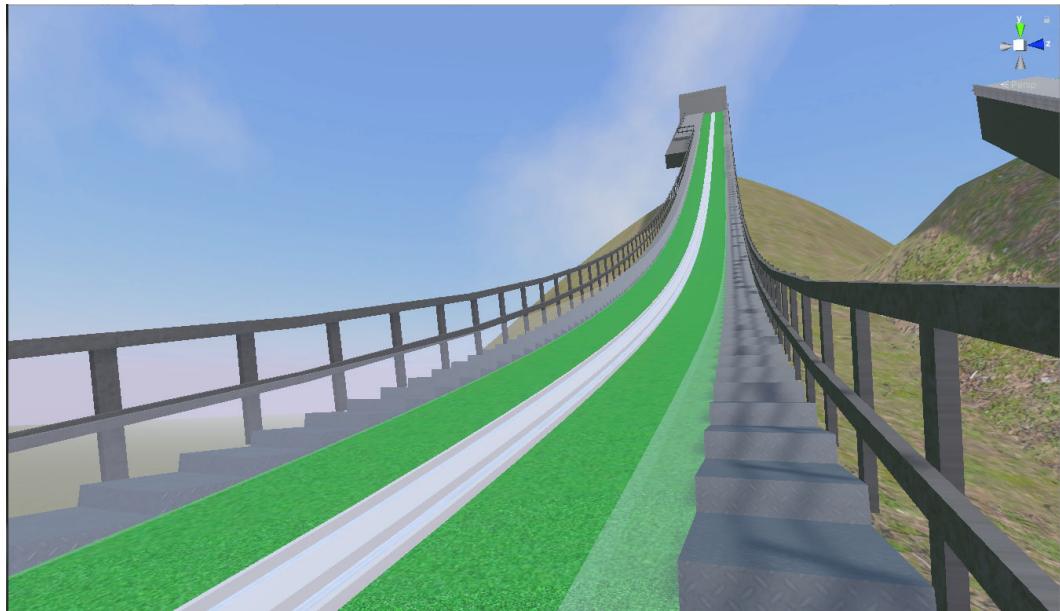
2.5. Obiekty proceduralne

Oprócz krzywych, do wygenerowania geometrii potrzebne będą pewne obiekty proceduralne.

2.5.1. Construction

Podstawowym elementem generatora jest obiekt **Construction**, który powstaje poprzez wyciągnięcie czworokąta wzdłuż pewnej krzywej, którą dalej będziemy nazywać osią obiektu. Oprócz wyciągnięcia czworokąta wzdłuż krzywej, dla każdego z wierzchołków czworokąta można podać krzywą, która będzie reprezentować funkcję odsunięcia danego wierzchołka od osi obiektu $\mathbb{R} \rightarrow \mathbb{R}^2$. Dodatkowo, **Construction** pozwala na stworzenie kilku obiektów wzdłuż osi, z przerwami pomiędzy kolejnymi

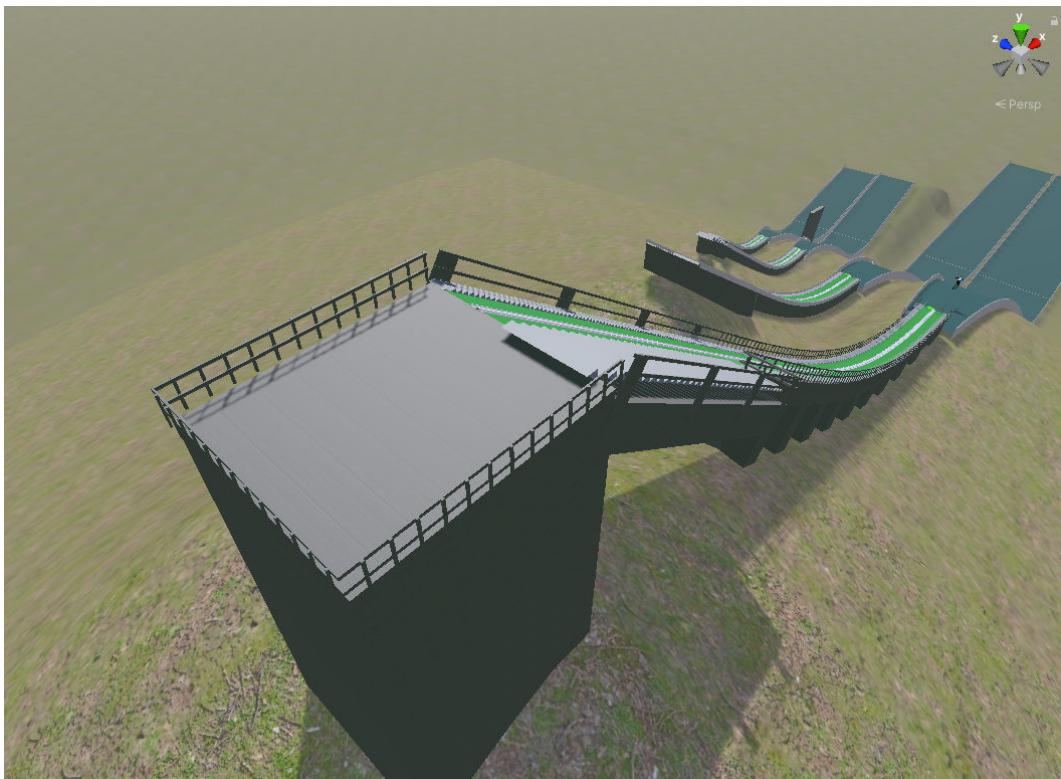
obiektami, zamiast jednego (w przypadku grafiki dwuwymiarowej, odpowiada to rysowaniu linii przerywanej, zamiast ciągłej).



Rysunek 2.6: Przykłady wygenerowanych barierek za pomocą obiektu **Construction**

W zależności od ustawienia osi obiektu, **Construction** można używać na kilka różnych sposobów. Pierwszą możliwością wykorzystania jest ustalenie kształtu obiektu za pomocą osi, a jako funkcje odsunięcia wierzchołków użycie funkcji stałych. Wykorzystując **Construction** w taki sposób można stworzyć między innymi: bandy, barierki, elementy konstrukcyjne skoczni, czy też drogi. Zostało to zaprezentowane na Rysunkach 2.6 oraz 2.7

Jednakże w przypadku generowania w taki sposób obiektów takich jak podpory, należy pamiętać, że nie będą one pionowe, lecz będą skierowane prostopadle do krzywej, która jest osią główną obiektu. Problem ten został zilustrowany na Rysunku 2.8

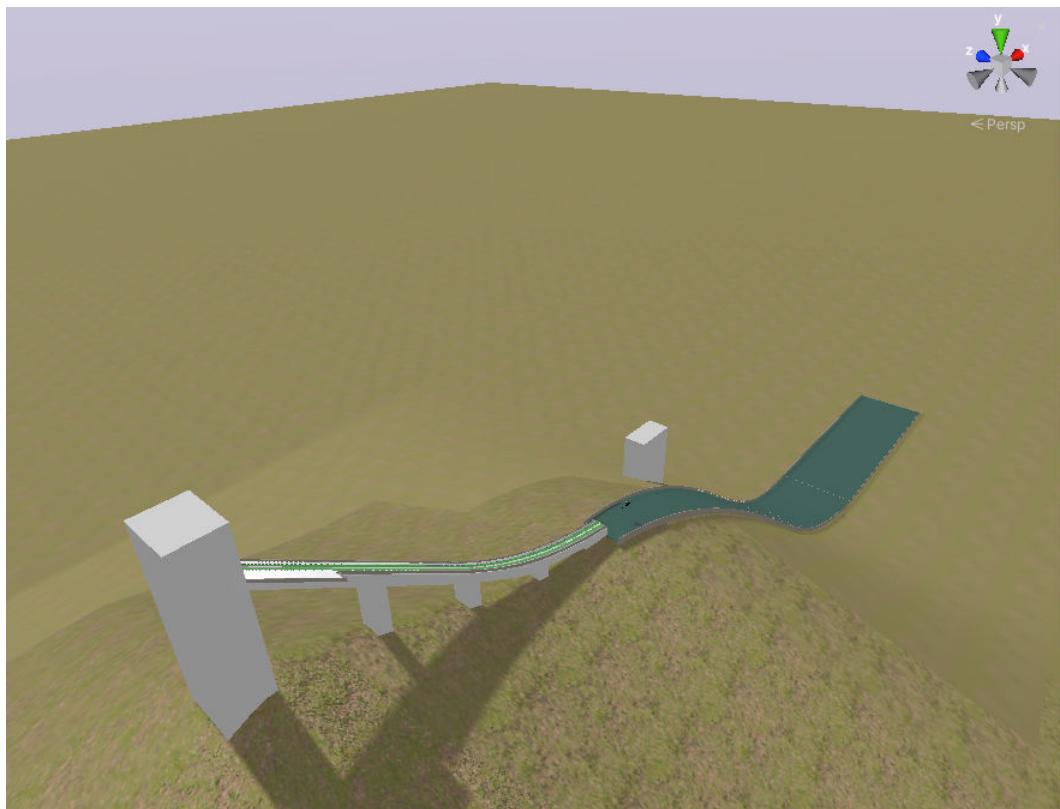


Rysunek 2.7: Przykłady wygenerowanych barierek za pomocą obiektu **Construction**



Rysunek 2.8: Przykład podpór wygenerowanych w obiekcie w którym osią główną jest krzywa najazdu, w wyniku czego podpory nie są poziome

Innym sposobem na wykorzystanie **Construction** może być ustawienie jako osi obiektu krzywej równoległej do osi skoczni i wykorzystanie funkcji odsunięcia wierzchołków do sterowania kształtem. Wówczas problem pokazany na Rysunku 2.8 można naprawić poprzez ustawienie krzywej najazdu jako funkcji odsunięcia dla górnych wierzchołków, a w przypadku wierzchołków dolnych - ustawienie krzywej w taki sposób, by podpory zaczynały się poniżej powierzchni terenu. Efekt takiego rozwiązania zostało pokazane na Rysunku 2.9



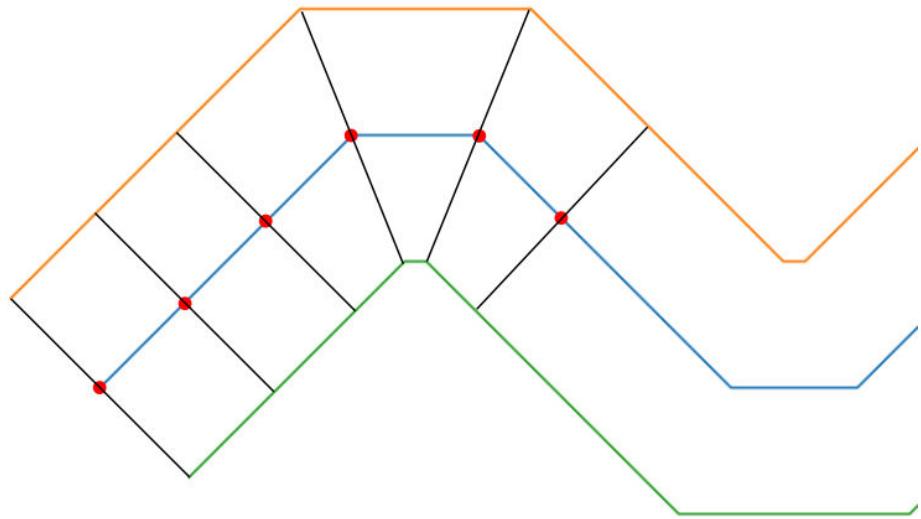
Rysunek 2.9: Podpory na skoczni wykonane według drugiego ze sposobów

Generator skoczni w przypadku obiektu **Construction** w pierwszej kolejności próbuje oś główną obiektu. Następnie, korzystając ze sposobu opisanego w sekcji o generowaniu krzywych równoległych, wyznacza wektory odsunięcia dla osi głównej w punktach, które zostały wcześniej spróbkowane.

2.5.2. Stairs

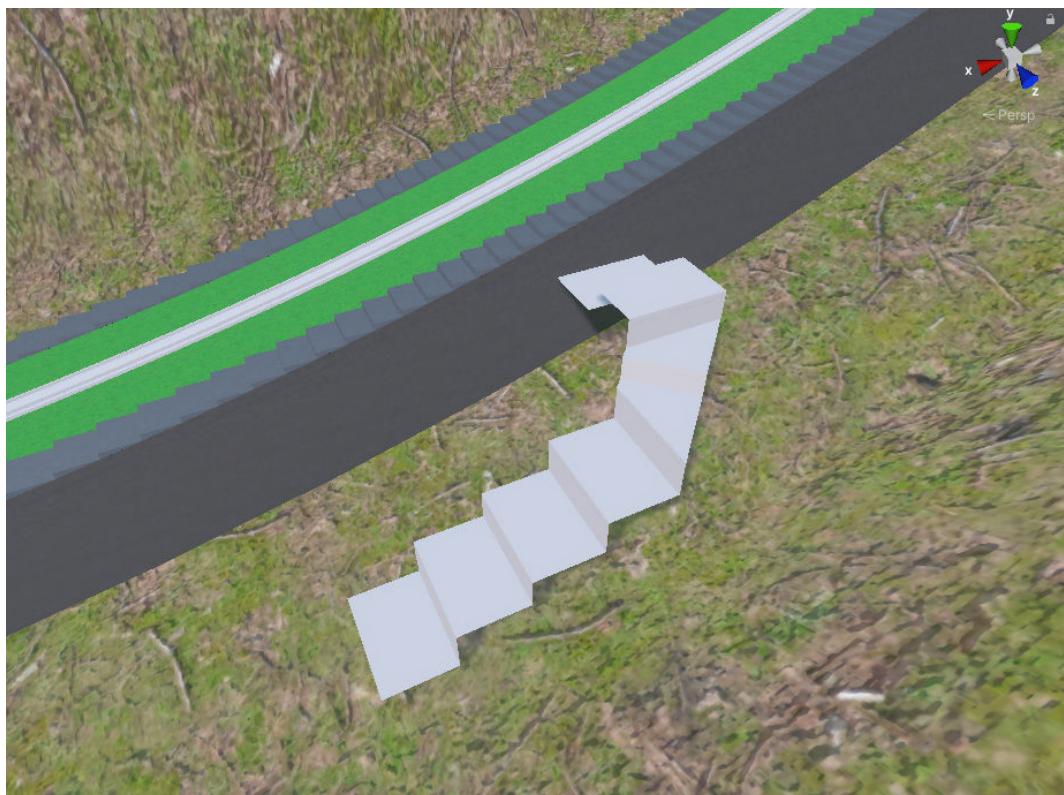
Kolejnym elementem, na którego generowanie pozwala generator skoczni jest obiekt **Stairs**, czyli schody. **Stairs** oprócz krzywej będącej osią główną obiektu, pozwala na ustawienie dwóch krzywych odsunięć wierzchołków, które w przeciwnieństwie do **Construction** będą reprezentować funkcję $\mathbb{R} \rightarrow \mathbb{R}$. W przypadku obiektu

Stairs, oś główna wyznacza kierunek prostopadły do stopni schodów. Oprócz krzywych, odpowiadających za kształt schodów, koniecznym jest ustalenie odległości, co jaką będą się tworzyły stopnie.



Rysunek 2.10: Ilustracja działania generatora schodów oraz przykład jego działania

Sposób generowania **Stairs**, jest analogiczny do **Construction** z tą różnicą, że krzywa jest próbkiwana w punktach, w których mają wypaść krawędzie stopni. Następnie, dla tych punktów obliczane są wektory przesunięć, dzięki którym otrzymujemy brzegi stopni. Jako, że płaszczyzna stopnia jest pozioma, możemy skorzystać z algorytmu obliczania krzywych równoległych dla przypadku dwuwymiarowego. Mając obliczone ułożenie stopni w rzucie poziomym, wykorzystując oś główną, będziemy mogli ustalić wysokości kolejnych stopni.



Rysunek 2.11: Przykład wygenerowanych schodów

30:94203

Rozdział 3.

Ukształtowanie terenu

Silnik Unity dostarcza gotowe rozwiązanie służące do modelowania terenu - klasę **Terrain** [8]. Oprócz wbudowanych w edytor narzędzi do modelowania terenu, klasa ta pozwala na manipulowanie właściwościami terenu za pomocą skryptów. Dodatkowo klasa **Terrain** pozwala na rozmieszczanie roślinności na powierzchni terenu, a także wspiera różne optymalizacje renderowania.

Skocznie narciarskie, które można spotkać w zawodach najwyższej rangi, są obiektami, które wymagają dosyć sporej różnicy wysokości. W związku z tym są one budowane zazwyczaj w taki sposób, by wykorzystać naturalne ukształtowanie terenu, choć zdarzają się obiekty, których konstrukcja wystaje znaczco ponad teren.

3.1. Dane wysokościowe

Możliwość wykorzystania rzeczywistych danych wysokościowych niewątpliwie może być pomocna w odwzorowaniu rzeczywistych skoczni narciarskich. Dane wysokościowe są dostępne w różnych formatach. Jednym z takich formatów jest hgt. W tym formacie jeden plik zawiera dane dla wszystkich punktów w obrębie kwadratu 1 stopień na 1 stopień, z rozdzielcością 3 sekund łuku. Taka rozdzielcość przekłada się na około 90 metrów odstępu pomiędzy sąsiednimi punktami, zatem konieczna jest interpolacja wysokości w punktach pośrednich. Jedną z klasycznych metod interpolacji w takich przypadkach jest interpolacja dwusześcienna.

Aby móc wykorzystać dane wysokościowe musimy najpierw dokonać przekształcenia z układu współrzędnych geograficznych do lokalnego układu współrzędnych, w którym opisywane są generowane skocznie. Aby uniknąć niepotrzebnych z punktu widzenia gry komplikacji obliczeń, będziemy przyjmować, że Ziemia jest kulą o promieniu R . W przypadku, gdy jako środek terenu na scenie przyjmiemy punkt w

układzie współrzędnych geograficznych $p = \begin{pmatrix} \lambda \\ \phi \\ h \end{pmatrix}$, gdzie λ to szerokość geograficzna,

ϕ to długość geograficzna, h to wysokość nad poziomem morza, płaszczyzną terenu będzie płaszczyzna styczna do kuli, która aproksymuje Ziemię, przechodzącą przez p . Płaszczyznę taką w skrócie będziemy określać jako LTP(p) (local tangent plane) [14].

Aby przekształcić współrzędne punktu z geograficznego układu współrzędnych do LTP, najpierw zajmiemy się przekształceniem punktu do układu współrzędnych ECEF (earth centered earth fixed). Środek tego układu to środek kuli ziemskiej, oś x skierowana do punktu $0^{\circ}\text{N } 0^{\circ}\text{E}$, oś y skierowana do punktu $0^{\circ}\text{N } 90^{\circ}\text{E}$, a oś z skierowana jest do północnego bieguna geograficznego.

$$\text{geogtoecef} \begin{pmatrix} \lambda \\ \phi \\ h \end{pmatrix} = (R + h) \begin{pmatrix} \cos \phi \cos \lambda \\ \sin \phi \\ \cos \phi \sin \lambda \end{pmatrix}$$

Przekształcenie do układu LTP wymaga jedynie stworzenia odpowiedniej macierzy obrotu:

$$\text{geogrotmat} = \text{rotmat}(-\phi, 0, 0) \cot \text{rotmat}(0, 90 + \lambda, 0),$$

a następnie przekształcenia za jej pomocą punktu z ECEF.

3.2. Modyfikacja ukształtowania terenu

Oprócz wykorzystania realnych danych wysokościowych, przydatna jest możliwość dowolnego kształtowania terenu. Jako że w Unity informacje o ukształtowaniu terenu są przechowywane w mapie wysokościowej - macierzy kwadratowej, to w przypadku terenu wokół skoczni, zadowalające efekty daje zastosowanie mapy jej rozdzielczości 1024×1024 , co przy zastosowaniu 16 bitowych zmiennych daje rozmiar około 2 MB, co znacząco zwiększyłoby rozmiar pliku opisującego skocznię, jak również wymagałoby to stworzenia dodatkowego do edycji.

3.2.1. Inverse distance weighting

Innym możliwym sposobem, na reprezentację ukształtowania terenu, może być zbiór punktów w których jest mierzona wysokość. Taki sposób wyznaczania ukształtowania terenu, wykorzystywany jest w geodezji, gdyż pomiary wysokości wykonuje się punktowo. Taka reprezentacja ukształtowania terenu wiąże się z koniecznością interpolacji wysokości w obszarach, w których wysokość nie została zmierzona. Jednym ze sposobów interpolacji jest metoda inverse distance weighting [13]. Zakłada ona, że wysokość w danym punkcie jest średnią ważoną wysokości w punktach, w których wysokość była mierzona, z wagą będącą odwrotnością odległości pomiędzy tymi punktami, podniesioną do pewnej potęgi.

Niech $p \in \mathbb{R}_+$ oznacza potęgę w wadze punktu, $u(\mathbf{x})$ oznacza wysokość terenu w punkcie \mathbf{x} . Dane mamy punkty w których były zmierzone wysokości \mathbf{x}_i dla $i = 1, 2, \dots, N$ oraz wartości wysokości w tych punktach $u_i = u(\mathbf{x}_i)$ dla $i = 1, 2, \dots, N$. Wówczas:

$$u(\mathbf{x}) = \begin{cases} \frac{\sum_{i=1}^N w_i(\mathbf{x}) u_i}{\sum_{i=1}^N w_i(\mathbf{x})}, & \text{jeżeli } d(\mathbf{x}, \mathbf{x}_i) \neq 0 \text{ dla wszystkich } i, \\ u_i, & \text{jeżeli } d(\mathbf{x}, \mathbf{x}_i) = 0 \text{ dla pewnego } i, \end{cases} \quad (3.1)$$

gdzie

$$w_i(\mathbf{x}) = \frac{1}{d(\mathbf{x}, \mathbf{x}_i)^p} \quad (3.2)$$

3.2.2. Modyfikacja metody inverse distance weighting

W przypadku w którym nie chcemy odwzorowywać rzeczywistego uksztaltowania powierzchni, lecz chcemy uksztaltować teren wedle naszego upodobania, możemy wprowadzić dodatkowe parametry dla węzłów które pozwolą na bardziej precyzyjne ustalenie uksztaltowania terenu. Zamiast jednej wartości p stałej dla wszystkich punktów, możemy ustalić osobno wagi dla każdego punktu pomiarowego. Oprócz danych jak w przypadku klasycznego IDW, dodatkowo mamy dane $p_i \in \mathbb{R}_+$ dla $i = 1, 2, \dots, N$. Wzór na $u(\mathbf{x})$ będzie taki sam jak 3.1. W przypadku modyfikacji IDW nie będzie można jednak zastosować 3.2, gdyż p nie jest jednakowe dla wszystkich punktów. Jednakże modyfikacja wzoru na w_i będzie bardzo nieznaczna:

$$w_i(\mathbf{x}) = \frac{1}{d(\mathbf{x}, \mathbf{x}_i)^{p_i}} \quad (3.3)$$

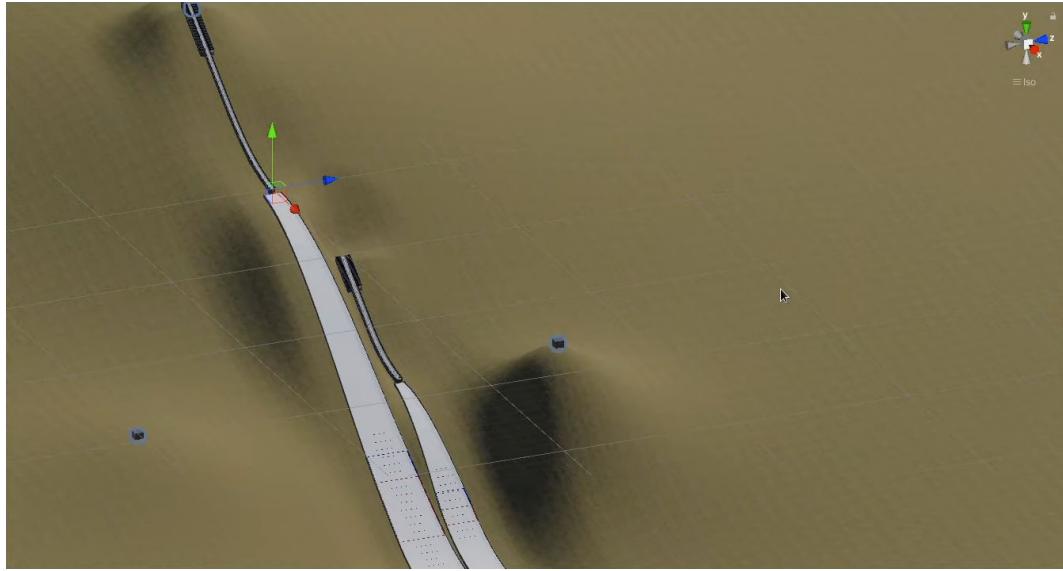
3.3. Dopasowanie uksztaltowania terenu do skoczni

W poniższych rozważaniach jako skocznę, będziemy rozumieć zbiór punktów leżących w obrębie rzutu poziomego skoczni.

3.3.1. Pojedyncza skocznia

Oprócz samego uksztaltowania terenu, potrzebna będzie metoda na dopasowanie uksztaltowania terenu w pobliżu skoczni. W przypadku jednej skoczni naturalnej, możemy ustalić pewną odległość do której skocznia będzie oddziaływała na teren d_0 . Dodatkowo potrzebna będzie funkcja obliczająca odległość punktu na płaszczyźnie od najbliższego punktu na skoczni. Mając obliczoną odległość punktu \mathbf{x} od skoczni h , dzielimy tę odległość przez d_0 uzyskując wagę dla tego punktu:

$$w(\mathbf{x}) = \max \left(1 - \frac{d(\mathbf{x}, h)}{d_0}, 0 \right) \quad (3.4)$$



Rysunek 3.1: Przykład wykorzystania zmodyfikowanej metody IDW w generowaniu terenu wokół skoczni

Idea tej metody została zaprezentowana na Rysunku 3.3

Niech $u(\mathbf{x})$ oznacza wysokość terenu w punkcie \mathbf{x} przed dopasowaniem terenu do skoczni, $g_h(\mathbf{x})$ oznacza wysokość terenu wokół skoczni h w punkcie na skoczni, który znajduje się najbliżej \mathbf{x} . Wówczas, ostateczna wysokość terenu w \mathbf{x} to:

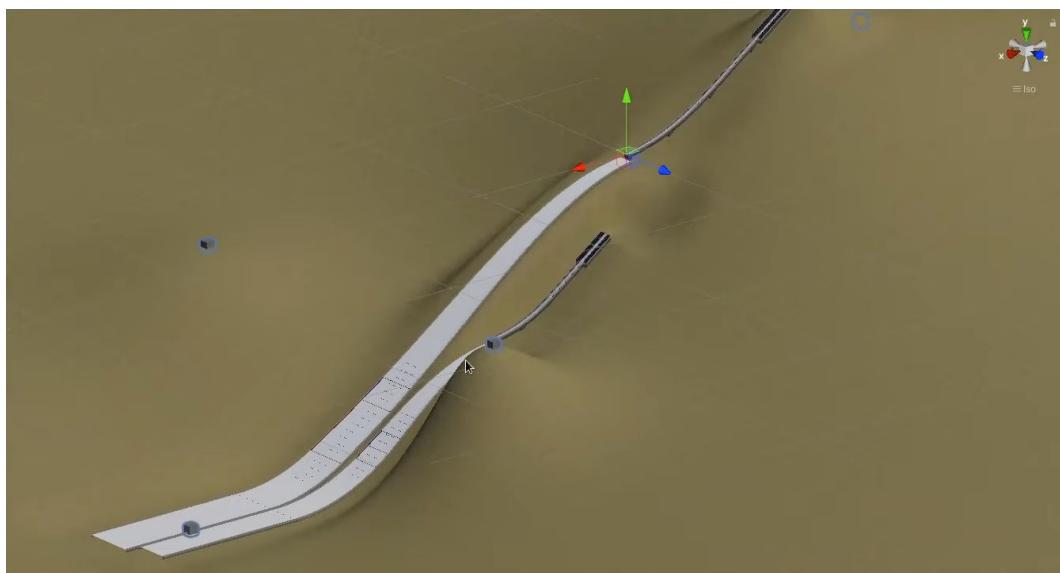
$$v(\mathbf{x}) = \text{smoothstep}(w(\mathbf{x}))g_h(\mathbf{x}) + \text{smoothstep}(1 - w(\mathbf{x}))u(\mathbf{x})$$

W przypadku skoczni ze sztuczną konstrukcją najazdową możemy wprowadzić dodatkowy parametr odpowiadający za przyleganie terenu do profilu najazdu - **terrainSteepness**. Działanie tego parametru zostało szczegółowo opisane we wstępie. Jeśli założymy, że osь skoczni jest osią x , próg skoczni znajduje się w punkcie $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ inrun(x) oznacza wysokość najazdu w punkcie x , a s oznacza wysokość progu skoczni, wówczas wysokość terenu w tym punkcie wyniesie:

$$t(x) = \text{terrainSteepness} \cdot (\text{inrun}(x) - s)$$

W przypadku, gdy parametr **terrainSteepness** wynosi 0, teren wokół rozbiegu wyrównywany jest do poziomu początku zeskoku, gdy parametr ten wynosi 1, wówczas teren jest wyrównywany do rozbiegu skoczni.

Dodatkowo, zamiast samego profilu skoczni, można dopasować teren do dowolnej krzywej ustalonej przez użytkownika, co pozwala na większą elastyczność generatora.

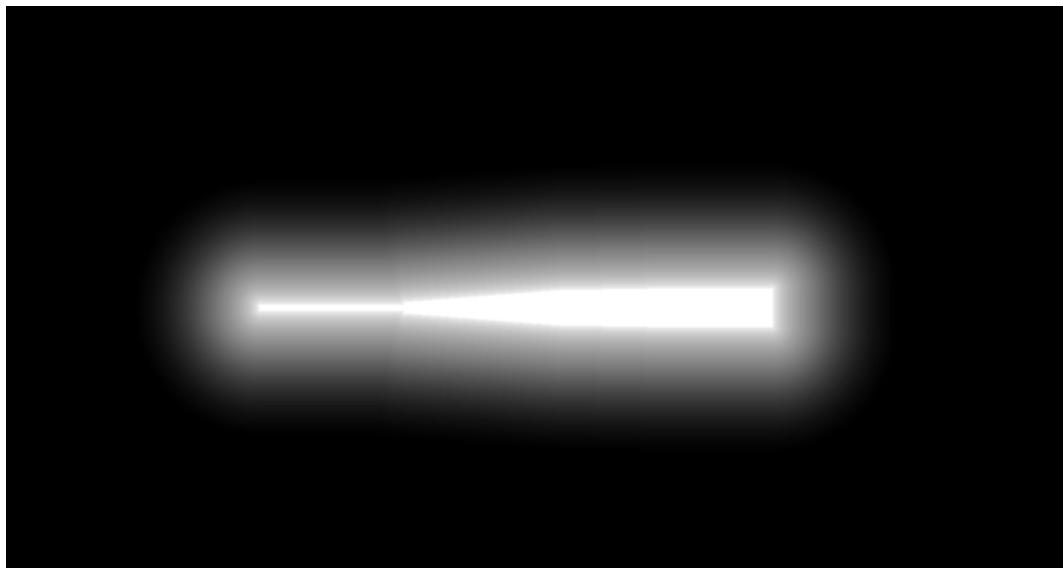


Rysunek 3.2: Przykład wykorzystania zmodyfikowanej metody IDW w generowaniu terenu wokół skoczni

3.3.2. Kompleksy skoczni

W rzeczywistości, skocznie stosunkowo często są częścią kompleksu składającego się z kilku skoczni. Naturalnie nasuwającym się rozwiązaniem jest wykonanie dopasowania terenu do każdej skoczni z osobna metodą dla pojedynczej skoczni. W przypadku, gdy skocznie są od siebie znacznie oddalone, takie podejście może przynieść zadowalający efekt. Niestety w prawdziwych kompleksach skoczni, skocznie są zazwyczaj umieszczone bardzo blisko siebie, zatem dopasowanie terenu do jednej skoczni, może sprawić, że inne skocznie będą zakryte, bądź będą wystawać nad powierzchnią terenu.

Jednym z możliwych rozwiązań tego problemu, jest wybranie dla każdego punktu skoczni znajdującej się najbliżej niego, i dopasowanie terenu w tym punkcie do tej skoczni. Niestety, jako że w Unity ukształtowanie terenu jest reprezentowane za pomocą mapy wysokości, wysokość terenu możemy ustalać z rozdzielczością około 1 metra. W takim przypadku, w punktach, które leżą w podobnej odległości do dwóch skoczni, mogą się pojawiać artefakty takie jak pokazane na Rysunku 3.5.

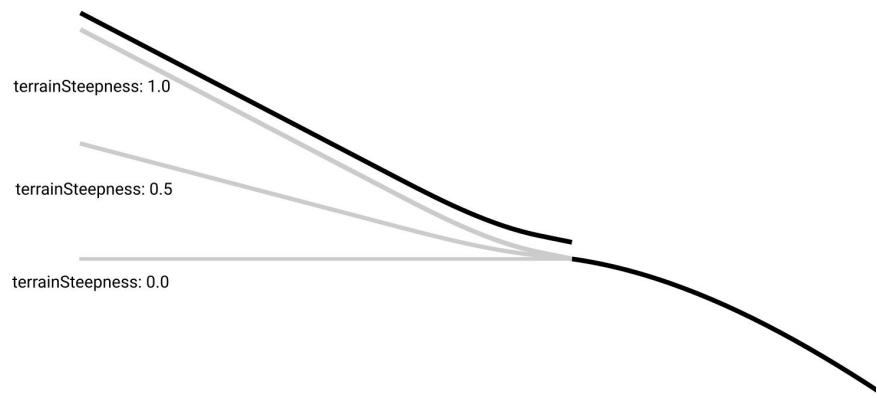


Rysunek 3.3: Mapa oddziaływania skoczni na teren wokół niej



Rysunek 3.5: Artefakty pojawiające się na terenie w przypadku zastosowania metody wyboru najbliższej skoczni

Rozwiązanie dające zadowalające efekty będzie bardzo podobne do metody IDW. Podobnie jak w przypadku tej metody, potrzebny będzie parametr p będący wykładnikiem potęgi przy obliczaniu wagi punktu. Dla punktu \mathbf{x} oraz dla każdej skoczni h_1, h_2, \dots, h_m , oraz funkcję wysokości skoczni h w punkcie \mathbf{x} , $g_h(\mathbf{x})$ oblicz



Rysunek 3.4: Różne ukształtowanie terenu wokół najazdu w zależność od parametru **terrainSteepness**

czamy:

$$\begin{aligned} a_i(\mathbf{x}) &= \min_{\mathbf{p} \in h_i}(d(\mathbf{p}, \mathbf{x})) \\ b_i(\mathbf{x}) &= g_{h_i}(a_i(\mathbf{x})) \\ w_i(\mathbf{x}) &= \frac{1}{d(a_i, \mathbf{x})^p} \end{aligned}$$

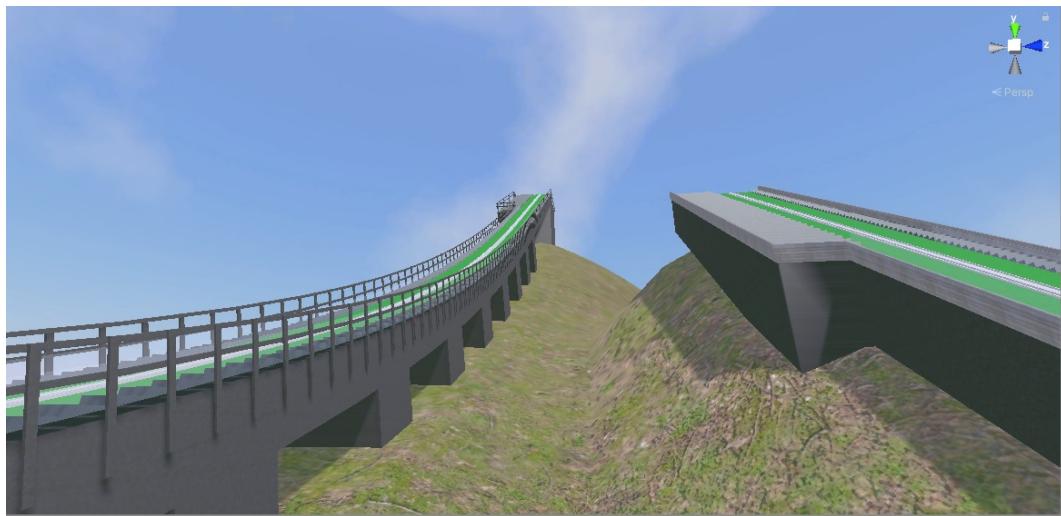
Dodatkowym parametrem, będzie q , czyli odległość do której skocznia wpływa na ukształtowanie terenu wokół niej. Przy założeniu, że wysokość terenu przed dopasowaniem go do skoczni to $u(\mathbf{x})$:

$$e_i(\mathbf{x}) = \text{Lerp}(b_i(\mathbf{x}), u(\mathbf{x}), \text{clamp01}(b_i(\mathbf{x})))$$

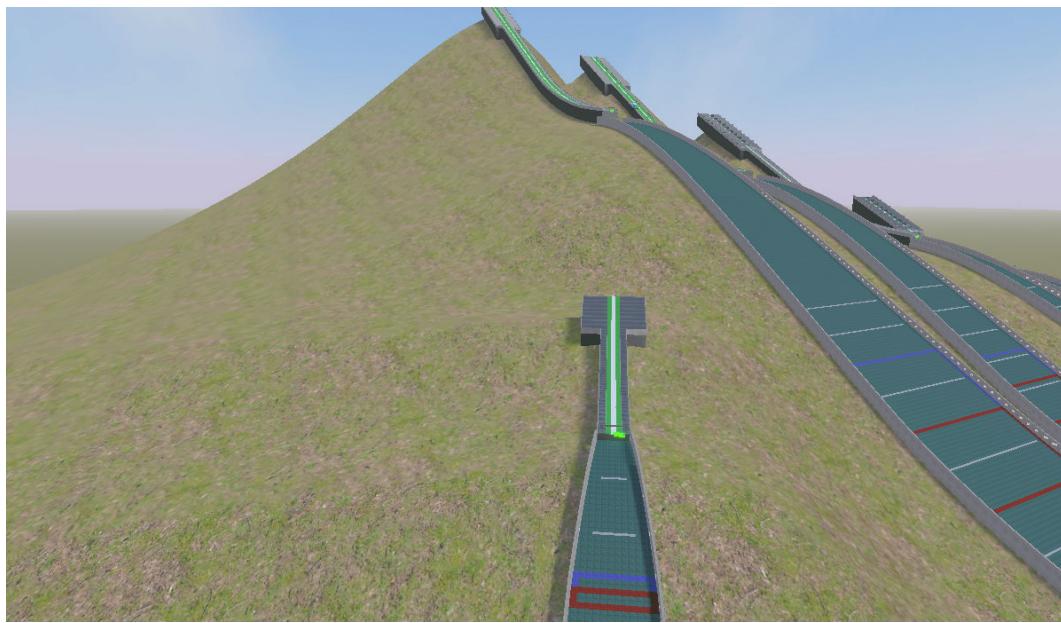
gdzie $\text{clamp01} = \max(\min(b_i(x), 1), 0)$.

Wówczas wysokość terenu to:

$$v(\mathbf{x}) = \begin{cases} \frac{\sum_{i=1}^N w_i(\mathbf{x}) e_i}{\sum_{i=1}^N w_i(\mathbf{x})}, & \text{jeżeli } d(\mathbf{x}, a_i) \neq 0 \text{ dla wszystkich } i, \\ e_i, & \text{jeżeli } d(\mathbf{x}, a_i) = 0 \text{ dla pewnego } i \end{cases} \quad (3.5)$$



Rysunek 3.6: Teren wygenerowany za pomocą drugiej metody - widoczny brak charakterystycznych artefaktów



Rysunek 3.7: Teren wygenerowany za pomocą drugiej metody - widoczny brak charakterystycznych artefaktów

Rozdział 4.

Geometria skoczni narciarskiej

Przy opisie proceduralnego generowania skoczni, pominięte zostały szczegółowe opisy profilu skoczni - wszelkie opisy zakładały, że znamy profil skoczni i potrafimy obliczyć krzywą reprezentującą najazd oraz zeskok skoczni. Jednakże poprawne wygenerowanie profilu jest zagadnieniem nietrywialnym, w związku z tym temat ten będzie opisany nieco bardziej szczegółowo, zważywszy na to, iż nie ma zbyt wielu źródeł traktujących ten temat.

4.1. Klasyfikacja skoczni narciarskich

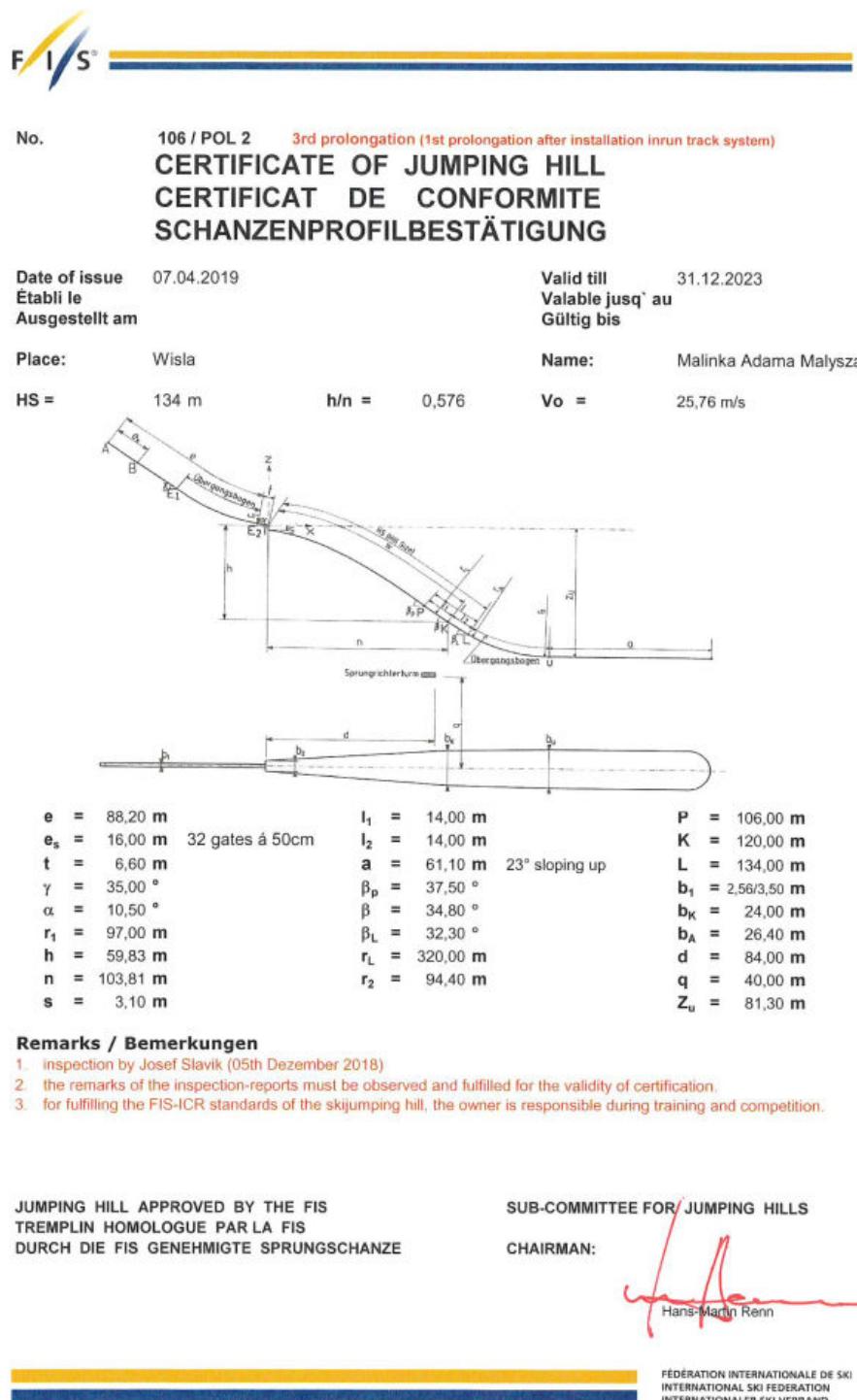
Skocznie narciarskie można sklasyfikować według ich rozmiaru. Rozmiar skoczni ma przede wszystkim wpływ na to, ile punktów otrzymuje zawodnik za odległość. Dodatkowo w zależności od rozmiaru skoczni przepisy nakładają więzy na niektóre parametry skoczni, dlatego też pogląd na to, jak wygląda klasyfikacja skoczni narciarskich według FIS, może się okazać pomocny w dalszej części tego rozdziału [5].

Rodzaj skoczni	HS (rozmiar skoczni)
Skocznia mała	co najwyżej 49 metrów
Skocznia średnia	od 50 do 84 metrów
Skocznia normalna	od 85 do 109 metrów
Skocznia duża	od 110 do 184 metrów
Skocznia do lotów narciarskich	co najmniej 185 metrów

4.2. Parametry profilu skoczni narciarskiej

Każda skocznia narciarska na której są organizowane międzynarodowe zawody musi posiadać ważną homologację Międzynarodowej Federacji Narciarskiej. Homologacja skoczni zawiera opis najważniejszych parametrów odpowiedzialnych za kształt

profilu skoczni. Homologacje skoczni certyfikowanych przez FIS są dostępne na ich stronie internetowej [6]. Przykład takiego certyfikatu znajduje się na Rysunku 4.1.



Rysunek 4.1: Certyfikat profilu skoczni narciarskiej w Wiśle Malince (źródło: [6])

Parametry odpowiadające za profil najazdu skoczni[1]:

- e - długość rozbiegu od najwyższego punktu startowego do początku progu
- e_s - długość rozbiegu od najwyższego punktu startowego do najniższego punktu startowego
- t - długość progu
- γ - nachylenie prostej części najazdu
- α - nachylenie progu
- r_1 - promień krzywizny krzywej przejściowej najazdu mierzony na początku progu
- b_1 - szerokość części najazdu po której zjeźdża zawodnik, w szczególności schody znajdujące się z boku najazdu nie wliczają się do tej szerokości

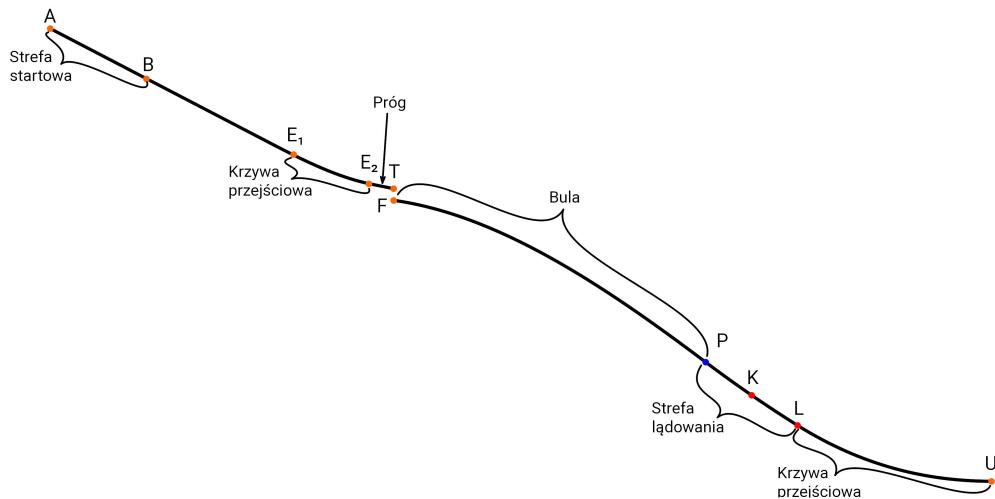
Parametry odpowiadające za profil zeskoku skoczni narciarskiej:

- h - odległość w pionie pomiędzy końcem progu, a punktem K
- n - odległość w poziomie pomiędzy końcem progu, a punktem K
- s - wysokość progu
- l_1 - długość zeskoku pomiędzy P i K
- l_2 - długość zeskoku pomiędzy K i L
- a - długość odjazdu - części skoczni w której zawodnicy hamują
- β_P - nachylenie zeskoku w punkcie P
- β - nachylenie zeskoku w punkcie K
- β_L - nachylenie zeskoku w punkcie L
- β_U - nachylenie zeskoku w punkcie U (w przypadku skoczni co najwyższej dużych, jest to 0, natomiast w przypadku skoczni do lotów narciarskich informację o tym parametrze uzyskamy z adnotacji umieszczonej przy parametrze a np. 4° sloping down after "U")
- r_L - promień łuku P-K-L
- r_{2L} - promień krzywej L-U mierzony w L
- r_2 - promień krzywej L-U mierzony w U
- b_2 - szerokość zeskoku za progiem

- b_K - szerokość zeskoku w punkcie K
- b_U - szerokość zeskoku w punkcie U

Punkty charakterystyczne skoczni:

- **A** - najwyższy możliwy punkt startowy na najeździe
- **B** - najniższy możliwy punkt startowy na najeździe
- **E₁** - początek krzywej przejściowej najazdu
- **E₂** - koniec krzywej przejściowej najazdu, początek progu
- **T** - koniec progu
- **F** - początek zeskoku (nie jest ujęty w przepisach)
- **P** - koniec buli, początek strefy lądowania (punkt przegięcia krzywej zeskoku)
- **K** - punkt konstrukcyjny skoczni, przeważnie znajduje się w okolicach środka strefy lądowania
- **L** - koniec strefy lądowania, początek krzywej przejściowej zeskoku
- **U** - koniec krzywej przejściowej zeskoku, na skoczniach dużych zeskok jest poziomy w tym punkcie, natomiast na skoczniach do lotów narciarskich, od kilku lat zezwala się, aby zeskok w tym punkcie był nachylony pod kątem co najwyżej 5 stopni



Rysunek 4.2: Ilustracja pokazująca rozmieszczenie punktów charakterystycznych oraz stref skoczni narciarskiej

4.3. Najazd

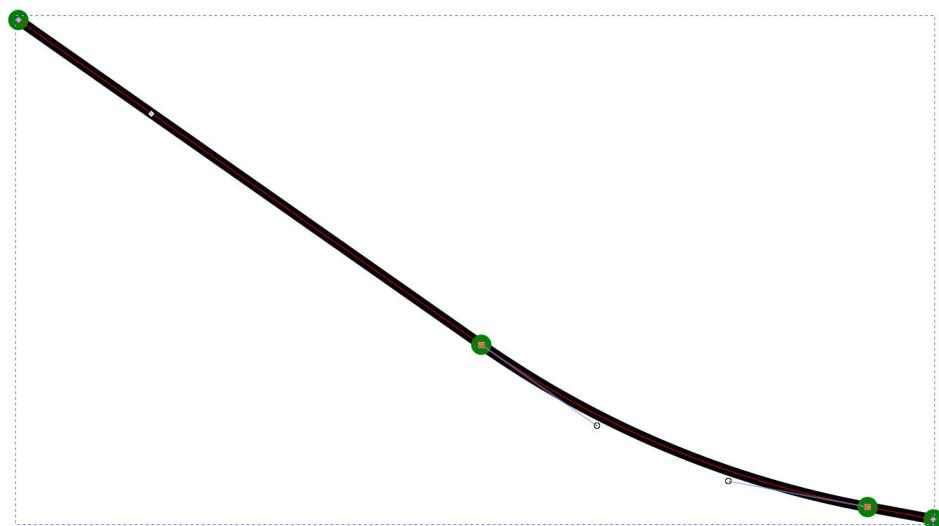
Najazd skoczni narciarskiej składa się z trzech stref:

- początkowej, o jednakowym nachyleniu γ
- krzywej przejściowej, mającej promień krzywizny w swoim końcu równy r_1
- progu o nachyleniu γ i długości t

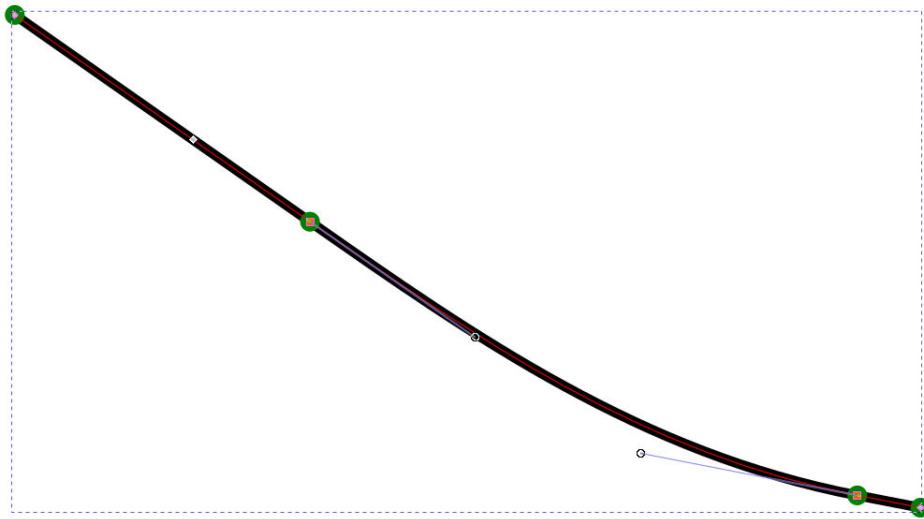
Ponadto e oznacza łączną długość dwóch pierwszych części najazdu. Początkową część pierwszej strefy o długości e_s nazywa się strefą startową. W tym obszarze skoczni przeważnie znajdują się schody wraz z miejscami w który można umieścić belkę startową, zwany punktami startowymi. Punkt początkowy najazdu, który jest jednocześnie jego najwyższym punktem startowym, oznaczamy jako **A**, natomiast najwyższy punkt startowy oznaczamy jako **B**. Oczywiście zachodzi $\|\mathbf{A} - \mathbf{B}\| = e_s$. Koniec pierwszej części, a zarazem początek krzywej przejściowej oznaczamy jako **E₁**, natomiast koniec krzywej przejściowej, czyli również początek progu oznaczamy jako **E₂**. Koniec progu, oznaczamy jako **T**, ponadto w tym punkcie będzie znajdująć się środka układu współrzędnych, który przyjmiemy do obliczeń.

W zależności od rodzaju profilu skoczni będziemy mieć do czynienia z różnymi możliwymi krzywymi przejściowymi. W przypadku **ICR1992** i **ICR1996** będzie to łuk (Rysunek 4.6), zaś w przypadku **ICR2008** - krzywa wielomianowa stopnia trzeciego (Rysunek 4.7).

Mając już wszystkie potrzebne dane, możemy przejść do obliczania współrzędnych punktów, które wykorzystamy do wyznaczania krzywej najazdu.



Rysunek 4.3: Geometria najazdu typu **ICR1992** oraz **ICR1996** (łuk został zastąpiony przez program Inkscape swoją przybliżoną reprezentacją krzywą Bezier)



Rysunek 4.4: Geometria najazdu typu ICR2008

4.3.1. Obliczenia

Próg ma długość t i jest nachylony pod kątem α , zatem początek progu \mathbf{E}_2 wyrazi się jako:

$$\mathbf{E}_2 = \mathbf{T} + t \begin{pmatrix} -\cos \alpha \\ \sin \alpha \end{pmatrix}$$

Ze względu na to, że w zależności od rodzaju krzywej przejściowej, otrzymamy różne wzory na \mathbf{E}_1 , zajmijmy się teraz obliczeniem \mathbf{A} i \mathbf{B} , zakładając, że znamy \mathbf{E}_1 , a także długość krzywej przejściowej, którą oznaczymy jako l . Wiemy, że \mathbf{E}

$$\|\mathbf{E}_2 - \mathbf{A}\| = e - l,$$

zatem:

$$\mathbf{A} = \mathbf{E}_1 + (e - l) \begin{pmatrix} -\cos \gamma \\ \sin \gamma \end{pmatrix}$$

Ponadto z tego, że:

$$e_s = \|\mathbf{A} - \mathbf{B}\|,$$

mamy:

$$\mathbf{B} = \mathbf{A} - e_s \begin{pmatrix} -\cos \gamma \\ \sin \gamma \end{pmatrix}$$

Teraz przejdziemy do obliczenia \mathbf{E}_1 . Zaczniemy od prostszego przypadku, w którym to krzywa przejściowa jest łukiem, a więc ma jednakową krzywiznę. Wprowadźmy punkt pomocniczy \mathbf{C}_1 . Będzie on oznaczał środek okręgu, do którego należy ten łuk:

$$\mathbf{C}_1 = \mathbf{E}_2 + r_1 \begin{pmatrix} \sin \alpha \\ \cos \alpha \end{pmatrix},$$

wówczas:

$$\mathbf{E}_1 = \mathbf{C}_1 - r_1 \begin{pmatrix} \sin \gamma \\ \cos \gamma \end{pmatrix}$$

W przypadku, w którym krzywa przejściowa jest krzywą wielomianową trzeciego stopnia (aproksymacją klotoidy), należy wówczas ją rozpatrywać w układzie współrzędnych z osiami obróconymi o kąt $-\gamma$ w stosunku do głównego układu współrzędnych, o środku położonym w \mathbf{E}_1 . Krzywa ta opisana w takim układzie współrzędnych wyrazi się jako:

$$f(u) = Cu^3, \quad (4.1)$$

jej pierwsza pochodna jako:

$$f'(u) = 3Cu^2,$$

natomiast druga:

$$f''(u) = 6Cu$$

Krzywizna tej krzywej wyrazi się jako:

$$k_f(u) = \frac{|f''(u)|}{(1 + (f'(u))^2)^{\frac{3}{2}}} = \frac{|6Cu|}{(1 + 9C^2u^4)^{\frac{3}{2}}} \quad (4.2)$$

Wiemy, że wykres funkcji f ma przechodzić przez punkt:

$$\mathbf{E}_2 = \begin{pmatrix} u_0 \\ v_0 \end{pmatrix},$$

dodatkowo:

$$f'(u_0) = \tan(\gamma - \alpha),$$

zatem:

$$\begin{aligned} f'(u_0) &= 3Cu_0^2 = \tan(\gamma - \alpha) \implies C = \frac{\tan(\gamma - \alpha)}{3u_0^2} \\ v_0 &= f(u_0) = \frac{1}{3} \tan(\gamma - \alpha)u_0 \end{aligned}$$

Wiemy, że promień krzywizny krzywej przejściowej w \mathbf{E}_2 ma wynosić r_1 , zatem:

$$k_f(u_0) = \frac{|f''(u_0)|}{(1 + (f'(u_0))^2)^{\frac{3}{2}}} = \frac{6u_0 \frac{\tan(\gamma - \alpha)}{3u_0^2}}{(1 + \tan^2(\gamma - \alpha))^{\frac{3}{2}}} = \frac{1}{r_1}$$

W związku z tym:

$$u_0 = \frac{2r_1 \tan(\gamma - \alpha)}{(1 + \tan^2(\gamma - \alpha))^{\frac{3}{2}}}$$

Jako że $\frac{\pi}{2} > \gamma - \alpha > 0$, to $\tan(\gamma - \alpha) = \frac{\sin(\gamma - \alpha)}{\cos(\gamma - \alpha)}$, więc:

$$1 + \tan^2 \theta = 1 + \frac{\sin^2 \theta}{\cos^2 \theta} = \frac{\cos^2 \theta + \sin^2 \theta}{\cos^2 \theta} = \frac{1}{\cos^2 \theta}$$

Stąd otrzymujemy, że:

$$u_0 = \frac{2r_1 \frac{\sin(\gamma-\alpha)}{\cos(\gamma-\alpha)}}{\left(\frac{1}{\cos^2(\gamma-\alpha)}\right)^{\frac{3}{2}}} = 2r_1 \sin(\gamma - \alpha) \cos^2(\gamma - \alpha)$$

Ostatecznie otrzymujemy:

$$\mathbf{E}_1 = \mathbf{E}_2 - \begin{pmatrix} \cos \gamma & \sin \gamma \\ -\sin \gamma & \cos \gamma \end{pmatrix} \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} \quad (4.3)$$

W przypadku krzywej przejściowej będącej łukiem, mamy już wszystkie dane potrzebne do uzyskania krzywej najazdu:

- Line(\mathbf{A} , \mathbf{E}_1)
- Arc(\mathbf{E}_1 , \mathbf{E}_2 , \mathbf{C}_1)
- Line(\mathbf{E}_2 , \mathbf{T})

Do narysowania krzywej przejściowej, która jest krzywą wielomianową stopnia trzeciego, będziemy mogli użyć krzywej Beziera trzeciego stopnia. Aby to uczynić należy wyznaczyć dwa punkty kontrolne. W przypadku używania krzywych Beziera trzeciego stopnia jako wielomianu aproksymacyjnego Hermite'a $h(x)$ dla punktów $A = (x_A, y_A)$ oraz $B = (x_B, y_B)$ oraz pochodnych k_1, k_2 , punkty kontrolne krzywej Beziera równoważnej $h(x)$ to:

$$C_1 = A + \frac{x_B - x_A}{3} \binom{1}{k_1}$$

$$C_2 = B - \frac{x_B - x_A}{3} \binom{1}{k_2}$$

W przypadku krzywej przejściowej rozbiegu, punkty te należy przekształcić do globalnego układu współrzędnych:

$$E_{1C} = E_1 + \frac{x_{E_2} - x_{E_1}}{3} \begin{pmatrix} \cos \gamma & \sin \gamma \\ -\sin \gamma & \cos \gamma \end{pmatrix} \begin{pmatrix} 1 \\ -\tan \gamma \end{pmatrix}$$

$$E_{2C} = E_2 - \frac{x_{E_2} - x_{E_1}}{3} \begin{pmatrix} \cos \gamma & \sin \gamma \\ -\sin \gamma & \cos \gamma \end{pmatrix} \begin{pmatrix} 1 \\ -\tan \alpha \end{pmatrix}$$

W przypadku krzywej przejściowej będącej krzywą wielomianową stopnia trzeciego, również mamy już wszystkie dane potrzebne do uzyskania krzywej najazdu:

- Line(\mathbf{A} , \mathbf{E}_1)
- Bezier3(\mathbf{E}_1 , \mathbf{E}_2 , \mathbf{E}_{1C} , \mathbf{E}_{2C})
- Line(\mathbf{E}_2 , \mathbf{T})

4.4. Zeskok

Zeskok skoczni narciarskiej składa się z trzech stref: buli, strefy lądownia i krzywej przejściowej.

4.4.1. Bula i strefa lądownia

Bula w przypadku każdego z rodzajów profilu jest krzywą wielomianową trzeciego stopnia, która przechodzi przez punkty \mathbf{F} i \mathbf{P} , w punkcie \mathbf{F} ma pochodną $-\tan \beta_0$, a w \mathbf{P} ma pochodną $-\tan \beta_P$.

Strefa lądownia w przypadku profilu **ICR1992** jest odcinkiem z \mathbf{P} do \mathbf{K} . W przypadku tego typu profilu punkt \mathbf{L} nie występował, gdyż końcem strefy lądownia był punkt \mathbf{K} , lecz dla uproszczenia obliczeń i kompatybilności z innymi typami profili, przyjmujemy $\mathbf{K} = \mathbf{L}$. Krzywa przejściowa w przypadku **ICR1992** jest łukiem o promieniu r_2 .

Począwszy od **ICR1996**, strefa lądownia przestała mieć jednakowe nachylenie, gdyż punkty \mathbf{P} , \mathbf{K} , \mathbf{L} leżą na łuku o promieniu r_L . W przypadku **ICR1996** krzywa przejściowa, podobnie jak w przypadku **ICR1992**, jest łukiem o promieniu r_2 .

W przypadku typu profilu **ICR2008**, strefa lądownia jest taka sama jak w przypadku **ICR1996**, zmianie uległa natomiast krzywa przejściowa, która od teraz jest parabolą o promieniu krzywizny r_{2L} w \mathbf{L} i r_2 w \mathbf{U} .

Z definicji mamy, że $\mathbf{K} = \begin{pmatrix} n \\ -h \end{pmatrix}$

W przypadku **ICR1992**, potrzebna nam jest jeszcze długość strefy lądownia, czyli M :

$$\mathbf{P} = \mathbf{K} - M \begin{pmatrix} -\cos \beta \\ \sin \beta \end{pmatrix}$$

$$\mathbf{L} = \mathbf{K}$$

Natomiast w przypadku, gdy strefa lądownia jest łukiem o promieniu r_L , potrzebne nam będą nachylenia stycznych w \mathbf{P} , \mathbf{K} oraz \mathbf{L} . Ponieważ mamy do czynienia z łukiem, wprowadzimy pomocniczy punkt \mathbf{C}_L , który będzie środkiem okręgu, do którego należy łuk przechodzący przez punkty \mathbf{P}, \mathbf{K} oraz \mathbf{L} .

$$\mathbf{C}_L = \mathbf{K} + r_L \begin{pmatrix} \sin \beta \\ \cos \beta \end{pmatrix}$$

Wówczas:

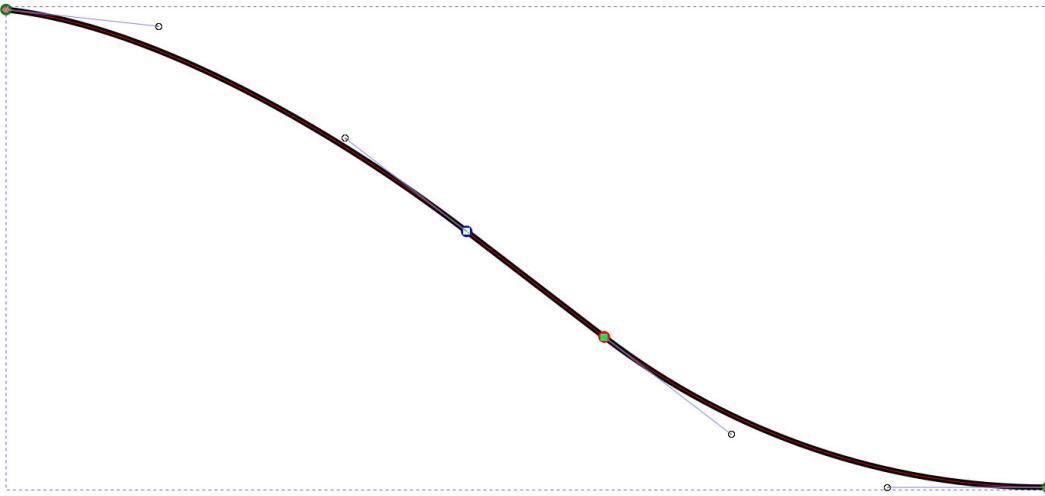
$$\mathbf{P} = \mathbf{C}_L - r_L \begin{pmatrix} \sin \beta_P \\ \cos \beta_P \end{pmatrix}$$

$$\mathbf{L} = \mathbf{C}_L - r_L \begin{pmatrix} \sin \beta_L \\ \cos \beta_L \end{pmatrix}$$

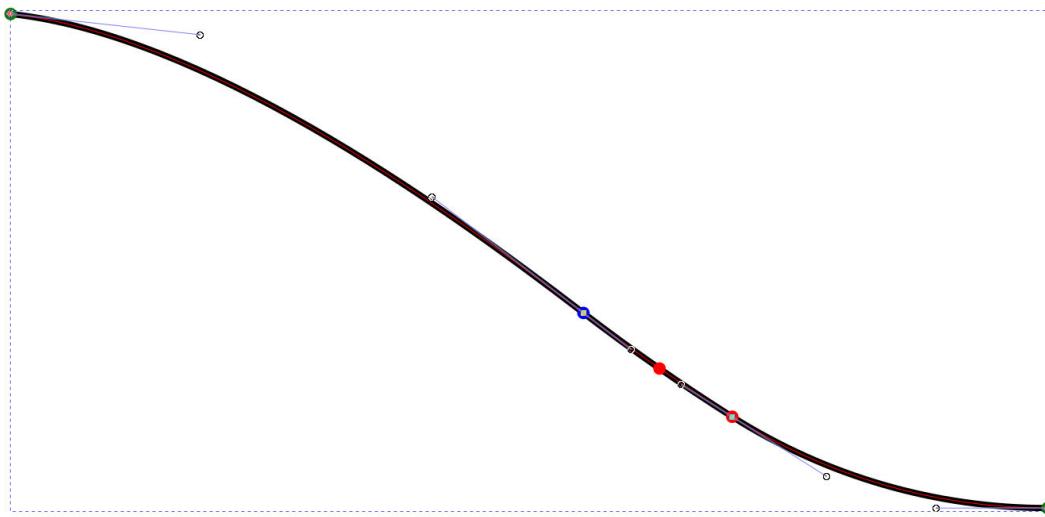
W przypadku buli, musimy wyznaczyć jedynie punkty kontrolne krzywej Beziera: F_C oraz P_C :

$$F_C = F + \frac{x_P - x_F}{3} \begin{pmatrix} 1 \\ -\tan \beta_0 \end{pmatrix}$$

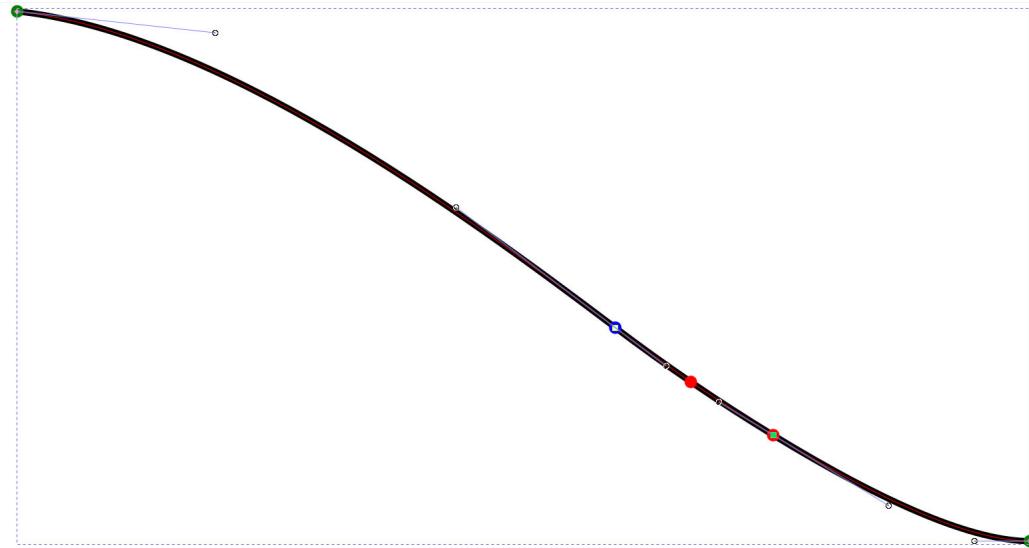
$$P_C = P - \frac{x_P - x_F}{3} \begin{pmatrix} 1 \\ -\tan \beta_P \end{pmatrix}$$



Rysunek 4.5: Geometria zeskoku typu **ICR1992** (łuki zostały zastąpione przez program Inkscape swoimi przybliżonymi reprezentacjami krzywymi Beziera)



Rysunek 4.6: Geometria zeskoku typu **ICR1996** (łuk został zastąpiony przez program Inkscape swoją przybliżoną reprezentacją krzywą Beziera)



Rysunek 4.7: Geometria zeskoku typu **ICR2008**

4.4.2. Krzywa przejściowa

W przypadku **ICR1992** oraz **ICR1996**, krzywa przejściowa jest łukiem o promieniu r_2 , wyznaczymy pomocniczy punkt \mathbf{C}_2 :

$$\mathbf{C}_2 = \mathbf{L} + r_2 \begin{pmatrix} \sin \beta_L \\ \cos \beta_L \end{pmatrix}$$

Wówczas koniec strefy lądowania to:

$$\mathbf{U} = \mathbf{C}_2 - r_2 \begin{pmatrix} \sin \beta_U \\ \cos \beta_U \end{pmatrix}$$

Mamy zatem wszystkie dane potrzebne do uzyskania krzywej zeskoku typu **ICR1992** oraz **ICR1996**. W przypadku **ICR1992** otrzymujemy:

- Bezier3($\mathbf{F}, \mathbf{P}, \mathbf{F}_C, \mathbf{P}_C$)
- Line(\mathbf{P}, \mathbf{L})
- Arc($\mathbf{L}, \mathbf{U}, \mathbf{C}_2$)

W przypadku **ICR1996** otrzymujemy:

- Bezier3($\mathbf{F}, \mathbf{P}, \mathbf{F}_C, \mathbf{P}_C$)
- Arc($\mathbf{P}, \mathbf{L}, \mathbf{C}_L$)
- Arc($\mathbf{L}, \mathbf{U}, \mathbf{C}_2$)

W przypadku typu profilu **ICR2008**, podobnie jak przy najeździe, obliczenie krzywej przejściowej będzie znaczowo bardziej skomplikowane. Jednakże zastosowanie krzywej o zmiennej krzywiznie sprawia, że przeciążenia działające na skoczka są mniejsze, bądź zmieniają się w sposób łagodniejszy niż w przypadku łuku. Parabola będzie obrócona, o kąt τ , którego z góry nie znamy. Podobnie środek paraboli, a zarazem środek układu współrzędnych nie jest znany. Wiemy natomiast, że krzywa ta będzie miała wzór $f(u) = Cu^2$, jej pierwszą pochodną będzie $f'(u) = 2Cu$, a drugą $f''(u) = 2C$. Krzywizna tej krzywej wyrazi się jako:

$$k_f(u) = \frac{|f''(u)|}{(1 + (f'(u))^2)^{\frac{3}{2}}} = \frac{|2C|}{(1 + 4C^2u^2)^{\frac{3}{2}}}$$

Punkty **L** oraz **U** w lokalnym układzie współrzędnych paraboli oznaczymy jako odpowiednio $\bar{\mathbf{L}}$ oraz $\bar{\mathbf{U}}$. Funkcja f będzie przechodziła przez punkt $\bar{\mathbf{L}}$ dla argumentu u_0 , natomiast przez $\bar{\mathbf{U}}$ dla argumentu u_1 , zatem:

$$\bar{\mathbf{L}} = \begin{pmatrix} u_0 \\ Cu_0^2 \end{pmatrix}, \bar{\mathbf{U}} = \begin{pmatrix} u_1 \\ Cu_1^2 \end{pmatrix}, \bar{\mathbf{S}} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Dodatkowo, możemy obliczyć pochodne w tych punktach:

$$\begin{aligned} f'(u_0) &= 2Cu_0 = -\tan(\tau + \beta_L) \\ f'(u_1) &= 2Cu_1 = -\tan(\tau + \beta_U) \end{aligned}$$

Znając promień krzywizny, otrzymamy:

$$\begin{aligned} k_f(u_0) &= \frac{|f''(u_0)|}{(1 + (f'(u_0))^2)^{\frac{3}{2}}} = \frac{2C}{(1 + \tan^2(\tau + \beta_L))^{\frac{3}{2}}} = \frac{1}{r_{2L}} \\ k_f(u_1) &= \frac{|f''(u_1)|}{(1 + (f'(u_1))^2)^{\frac{3}{2}}} = \frac{2C}{(1 + \tan^2(\tau + \beta_U))^{\frac{3}{2}}} = \frac{1}{r_2} \\ 2C &= \frac{(1 + \tan^2(\tau + \beta_L))^{\frac{3}{2}}}{r_{2L}} = \frac{(1 + \tan^2(\tau + \beta_U))^{\frac{3}{2}}}{r_2} \end{aligned}$$

Ponieważ r_{2L} oraz r_2 są dodatnie, możemy podzielić równanie stronami. Wówczas otrzymamy:

$$\begin{aligned} \frac{(1 + \tan^2(\tau + \beta_U))^{\frac{3}{2}}}{(1 + \tan^2(\tau + \beta_L))^{\frac{3}{2}}} &= \frac{r_2}{r_{2L}} \\ \frac{1 + \tan^2(\tau + \beta_U)}{1 + \tan^2(\tau + \beta_L)} &= \left(\frac{r_2}{r_{2L}} \right)^{\frac{2}{3}} \\ \frac{\cos^2(\tau + \beta_L)}{\cos^2(\tau + \beta_U)} &= \left(\frac{r_2}{r_{2L}} \right)^{\frac{2}{3}} \\ \frac{\cos(\tau + \beta_L)}{\cos(\tau + \beta_U)} &= \left(\frac{r_2}{r_{2L}} \right)^{\frac{1}{3}} \end{aligned}$$

Korzystając ze wzoru na cos sumy, otrzymamy:

$$\begin{aligned}\frac{\sin \tau \sin \beta_L - \cos \tau \cos \beta_L}{\sin \tau \sin \beta_U - \cos \tau \cos \beta_U} &= \left(\frac{r_2}{r_{2L}} \right)^{\frac{1}{3}} \\ (\sin \tau \sin \beta_L - \cos \tau \cos \beta_L) &= (\sin \tau \sin \beta_U - \cos \tau \cos \beta_U) \left(\frac{r_2}{r_{2L}} \right)^{\frac{1}{3}} \\ \sin \tau \left(\sin \beta_L - \sin \beta_U \left(\frac{r_2}{r_{2L}} \right)^{\frac{1}{3}} \right) &= \cos \tau \left(\cos \beta_L - \cos \beta_U \left(\frac{r_2}{r_{2L}} \right)^{\frac{1}{3}} \right)\end{aligned}$$

τ nie może być mniejsze niż $-\frac{\pi}{2}$ lub większe niż $\frac{\pi}{2}$, gdyż wówczas nie otrzymam krzywej przejściowej, która będzie funkcją malejącą i wypukłą. Stąd:

$$\begin{aligned}\tan \tau &= \frac{\cos \beta_L - \cos \beta_U \left(\frac{r_2}{r_{2L}} \right)^{\frac{1}{3}}}{\sin \beta_L - \sin \beta_U \left(\frac{r_2}{r_{2L}} \right)^{\frac{1}{3}}} \\ \tau &= \arctan \left(\frac{\cos \beta_L - \cos \beta_U \left(\frac{r_2}{r_{2L}} \right)^{\frac{1}{3}}}{\sin \beta_L - \sin \beta_U \left(\frac{r_2}{r_{2L}} \right)^{\frac{1}{3}}} \right)\end{aligned}$$

Następnie wyznaczamy pozostałe niewiadome:

$$\begin{aligned}C &= \frac{(1 + \tan^2(\tau - \beta_U))^{\frac{3}{2}}}{2r_2} = \frac{1}{2r_2(\cos^2(\tau - \beta_U))^{\frac{3}{2}}} = \frac{1}{2r_2 \cos^3(\tau - \beta_U)} \\ 2Cu_0 &= -\tan(\tau + \beta_L) \implies u_0 = -\frac{\tan(\tau + \beta_L)}{2C} \\ 2Cu_1 &= -\tan(\tau + \beta_U) \implies u_1 = -\frac{\tan(\tau + \beta_U)}{2C}\end{aligned}$$

Kolejnym krokiem będzie obliczenie położenia punktów charakterystycznych na skoczni:

$$\begin{aligned}\bar{\mathbf{L}} &= \begin{pmatrix} u_0 \\ Cu_0^2 \end{pmatrix}, \bar{\mathbf{U}} = \begin{pmatrix} u_1 \\ Cu_1^2 \end{pmatrix} \\ \mathbf{U} &= \mathbf{L} + \begin{pmatrix} \cos \tau & -\sin \tau \\ \sin \tau & \cos \tau \end{pmatrix} (\bar{\mathbf{U}} - \bar{\mathbf{L}}) = \mathbf{L} + \begin{pmatrix} \cos \tau & -\sin \tau \\ \sin \tau & \cos \tau \end{pmatrix} \begin{pmatrix} u_1 - u_0 \\ C(u_1^2 - u_0^2) \end{pmatrix} \\ \mathbf{S} &= \mathbf{L} + \begin{pmatrix} \cos \tau & -\sin \tau \\ \sin \tau & \cos \tau \end{pmatrix} (\bar{\mathbf{S}} - \bar{\mathbf{L}}) = \mathbf{L} - \begin{pmatrix} \cos \tau & -\sin \tau \\ \sin \tau & \cos \tau \end{pmatrix} \begin{pmatrix} u_0 \\ Cu_0^2 \end{pmatrix}\end{aligned}$$

Podobnie jak w przypadku buli, oraz krzywej przejściowej najazdu typu **ICR2008**, będziemy potrzebować obliczyć punkt kontrolny krzywej Beziera stopnia drugiego. Krzywa ta ma być równa fragmentowi paraboli, zatem punkt kontrolny \mathbf{U}_C znajduje się na przecięciu stycznych do paraboli w jej końcach.

$$\mathbf{U}_C = \mathbf{L} + t \begin{pmatrix} \cos \beta_L \\ -\sin \beta_L \end{pmatrix} = \mathbf{U} + v \begin{pmatrix} \cos \beta_U \\ -\sin \beta_U \end{pmatrix}$$

Niech $\mathbf{U} = \begin{pmatrix} x_U \\ y_U \end{pmatrix}$, $\mathbf{L} = \begin{pmatrix} x_L \\ y_L \end{pmatrix}$.

Rozwiążanie otrzymujemy korzystając ze wzorów na przecięcie prostych:

$$t = \frac{(x_L - x_U) \sin \beta_U + (y_L - y_U) \cos \beta_U}{\cos \beta_L \sin \beta_U - \sin \beta_L \cos \beta_U}$$

Podstawiając, otrzymujemy:

$$\mathbf{U}_C = \mathbf{L} + \frac{(x_L - x_U) \sin \beta_U + (y_L - y_U) \cos \beta_U}{\cos \beta_L \sin \beta_U - \sin \beta_L \cos \beta_U} \begin{pmatrix} \cos \beta_L \\ -\sin \beta_L \end{pmatrix}$$

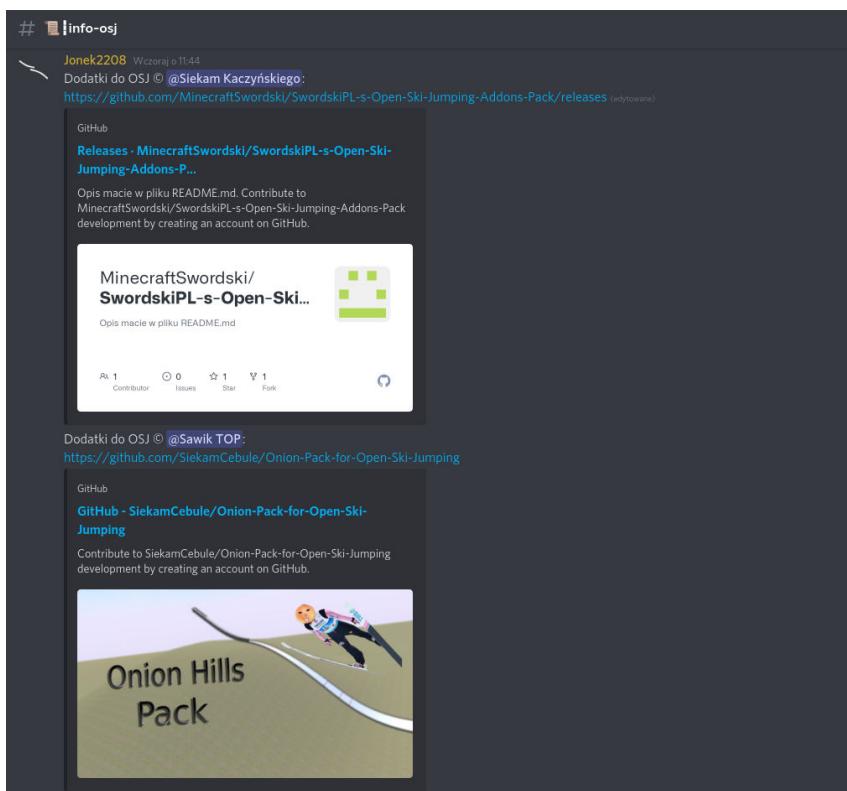
Krzywa najazdu w przypadku **ICR2008** jest następująca:

- Bezier3($\mathbf{F}, \mathbf{P}, \mathbf{F}_C, \mathbf{P}_C$)
- Arc($\mathbf{P}, \mathbf{L}, \mathbf{C}_L$)
- Bezier2($\mathbf{L}, \mathbf{U}, \mathbf{U}_C$)

Rozdział 5.

Podsumowanie

Rozwiązania przedstawione w pracy pozwoliły na poprawę wyglądu generowanych skoczni. W stosunku do możliwości jakie oferowały poprzednie wersje (stworzenie pojedynczej skoczni), teraz można stworzyć kilka skoczni na jednej mapie, jak również można dowolnie dopasować ich wygląd, czy też zmieniać ukształtowanie terenu. Gra pozwala również na odtworzenie rzeczywistego ukształtowania terenu w danym miejscu na Ziemi, korzystając z danych wysokościowych SRTM3.



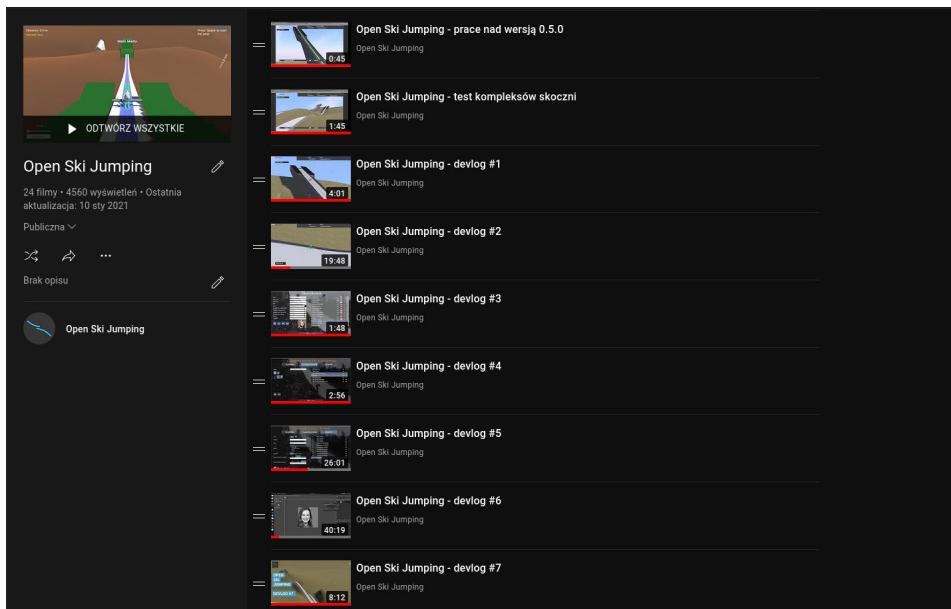
Rysunek 5.1: Dodatki zamieszczane przez graczy na serwerze gry na Discordzie

Niewątpliwym sukcesem projektu Open Ski Jumping jest to, że wokół gry utwo-

rzyła się społeczność fanowska. Osoby zainteresowane grą mogą dołączyć do serwera na Discordzie lub do grupy na Facebooku, na których inni członkowie chętnie służą pomocą graczom mającym kłopoty z grą. W tych serwisach gracze mogą również zgłaszać zaobserwowane w grze błędy. Mogą tam również zgłaszać swoje pomysły do kolejnych wersji, dzięki temu gra zyskała niektóre spośród swoich funkcji, takie jak możliwość dodania zdjęć zawodników. Niektórzy z graczy chętnie udostępniają innym swoje paczki skoczni, paczki zawodników, czy też pliki z konkursami i turniejami. Dzięki temu osoby, które nie są obeznane z systemem konkursów w grze mogą wykorzystać pliki stworzone przez osoby bardziej z nim obyty. Część graczy umieszcza swoje dodatki na GitHubie, dzięki czemu nabywa doświadczenie w używaniu systemów kontroli wersji.

Dzięki temu że można mieć wgląd w kod gry, niektórzy z członków społeczności (spośród których spora część to uczniowie) zdecydowały się na rozpoczęcie nauki programowania, czy też modelowania 3D. Oprócz samej gry Open Ski Jumping, na podstawie jej kodu źródłowego powstało kilka mniejszych projektów, takich jak symulator skoków narciarskich w konsoli (terminalu). Został on stworzony jako projekt na zaliczenie z przedmiotu Python 2. Jego kod źródłowy jest udostępniany na licencji MIT [16].

Oprócz stosunkowo sporego zainteresowania grą, jej rozwój przyczynił się do zwiększenia wiedzy na temat konstrukcji skoczni narciarskich wśród graczy, którzy wykorzystują tą wiedzę tworząc własne skocznie. Mając to na względzie, w pracy obliczenia profilu skoczni zostały opisane w sposób nieco bardziej szczegółowy.

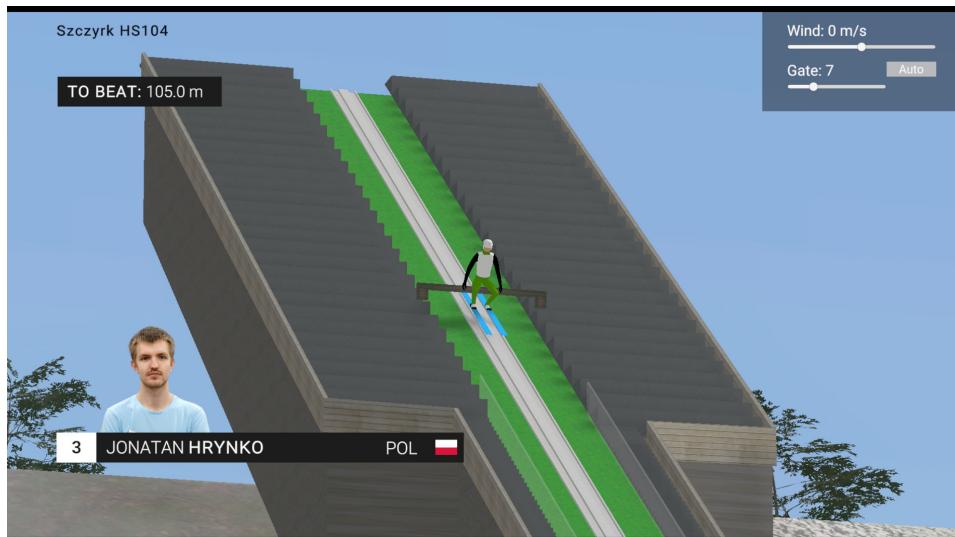


Rysunek 5.2: Oficjalny kanał Open Ski Jumping na YouTube

Na oficjalnym kanale gry w serwisie YouTube zamieszczane są filmy z postępami w grze, większość z nich ma przynajmniej kilka tysięcy wyświetleń. Kanał Open Ski

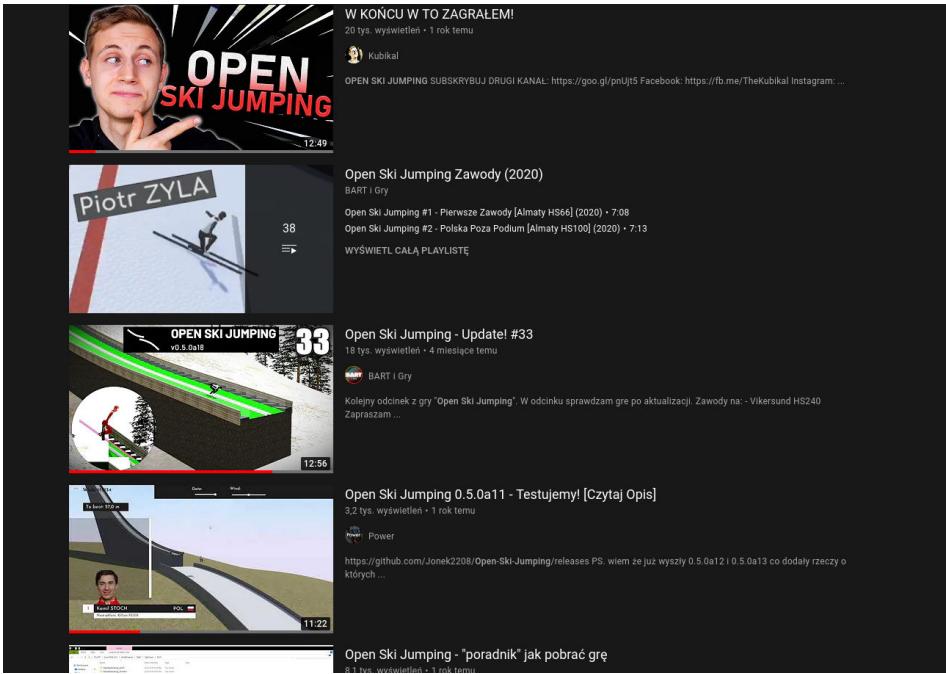
Jumping ma już prawie 3600 subskrybentów, a filmy na nim zostały wyświetlane łącznie ponad milion razy. Na kanale została stworzona playlista, która pokazuje rozwój gry od samego początku jej istnienia. Dzięki temu gracze mogą mieć wgląd w proces tworzenia gier komputerowych przez niezależnych twórców.

Gra pozwala graczom na dodawanie zdjęć, które potem mogą być wyświetlane na grafikach przed skokiem. Grafiki te swoim stylem nawiązują do grafik pokazywanych w transmisjach telewizyjnych z zawodów Pucharu Świata w skokach narciarskich.



Rysunek 5.3: Dodane do gry zdjęcia wyświetlane są podczas konkursów

Pozwala to graczom na lepsze wczucie się w klimat rywalizacji sportowej, który dotychczas był im znany tylko z transmisji telewizyjnych. Dzięki tym funkcjonalnościom, gracze mogą również tworzyć filmy na YouTube, które są symulacją prawdziwych zawodów skoków narciarskich. Filmy z gry zamieszczane są nie tylko na kanale Open Ski Jumping. Wielu twórców nagrywających filmy z gier o skokach narciarskich zamieszcza swoje filmy z gry Open Ski Jumping na swoich kanałach. Wyświetlenia niektórych z tych filmów dochodzą nawet do kilkudziesięciu tysięcy.



Rysunek 5.4: Filmy o grze Open Ski Jumping stworzone przez innych twórców

Dalszymi etapami rozwoju projektu, oprócz rozwoju generatora skoczni będzie między innymi dodanie możliwości rozgrywki sieciowej. W obecnych czasach sporo gier komputerowych zawiera taki tryb gry. Umożliwia to graczom konkurowanie ze sobą nawzajem, co w przypadku gry o tematyce sportowej ma szczególne znaczenie. Zmiany te w perspektywie czasu powinny przyczynić się do dalszego zwiększenia grona graczy.

W przyszłości planowane jest wydanie gry w serwisie Steam. Pozwoli to na łatwiejszy dostęp większej ilości osób, a także umożliwi dodanie do gry funkcji korzystających z API serwisu Steam, w celu dalszego urozmaicenia rozgrywki. Oprócz wersji na PC planowane jest także stworzenie wersji Open Ski Jumping na system Android.

Bibliografia

- [1] Hans-Heini Gasser, *Standards for the Construction of Jumping Hills - 2012*
- [2] Hans-Heini Gasser, *Grundlagen der Auslegung des Längsprofils einer Skisprungschanze*
- [3] Hans-Heini Gasser, *Grundlagen für die Projektierung einer Skisprungschanze*
- [4] Hans-Heini Gasser, *Computerprogramme Sprungschanzen Arbeitshilfen für die Anwendung von Art. 410 ff. der FIS IWO Band III*
- [5] https://assets.fis-ski.com/image/upload/v1626080270/fis-prod/assets/ICR_Ski_Jumping_2022_marked-up.pdf,
(dostęp: 30 sierpnia 2021)
- [6] <https://www.fis-ski.com/DB/ski-jumping/homologations.html>,
(dostęp: 30 sierpnia 2021)
- [7] <https://docs.unity3d.com/Manual/Example-CreatingaBillboardPlane.html>,
(dostęp: 30 sierpnia 2021)
- [8] <https://docs.unity3d.com/ScriptReference/TerrainData.html>,
(dostęp: 30 sierpnia 2021)
- [9] <https://docs.unity3d.com/ScriptReference/AnimationCurve.html>,
(dostęp: 30 sierpnia 2021)
- [10]
- [11] <https://docs.unity3d.com/ScriptReference/MeshFilter.html>,
(dostęp: 30 sierpnia 2021)
- [12] <https://docs.unity3d.com/ScriptReference/MeshRenderer.html>,
(dostęp: 30 sierpnia 2021)
- [13] https://en.wikipedia.org/wiki/Inverse_distance_weighting,
(dostęp: 30 sierpnia 2021)

- [14] https://en.wikipedia.org/wiki/Geographic_coordinate_system,
(dostęp: 30 sierpnia 2021)
- [15] <https://github.com/Jonek2208/Open-Ski-Jumping>,
(dostęp: 30 sierpnia 2021)
- [16] <https://github.com/ewintergames/sj-simulator>,
(dostęp: 30 sierpnia 2021)