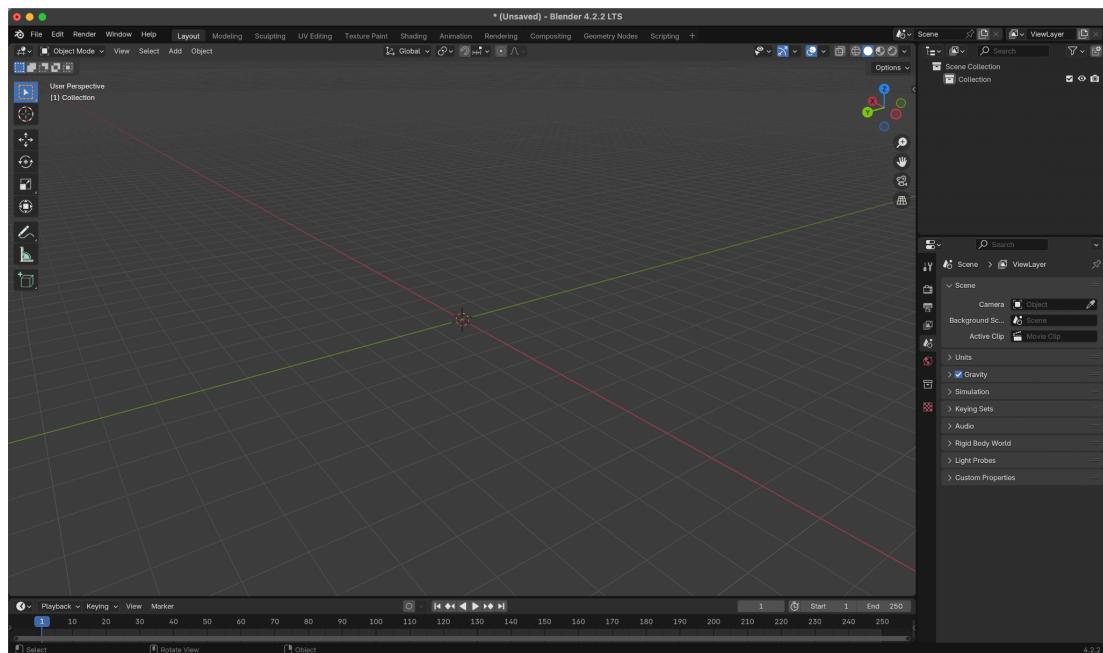


# Používateľská príručka

## Inštalácia

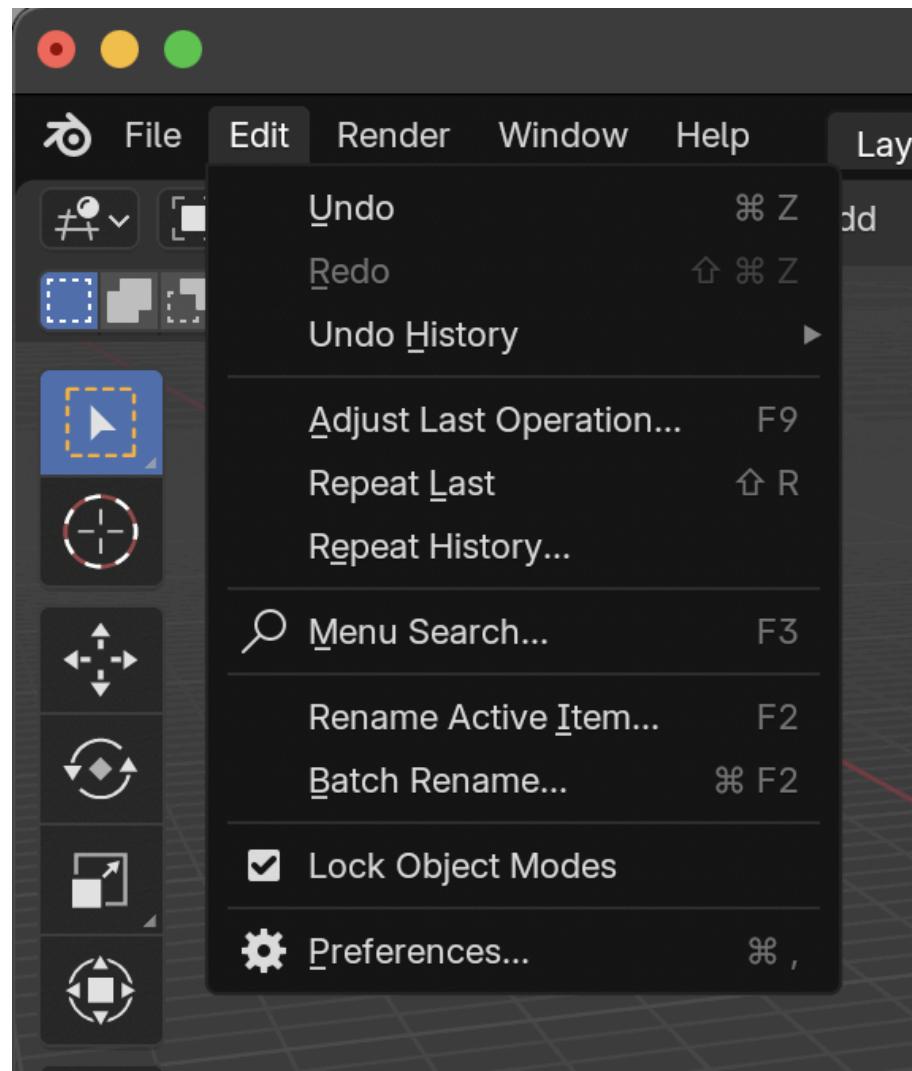
V prvom rade je dôležité mať stiahnutý a nainštalovaný Blender z oficiálneho zdroja [blender\_source]. Kedže toto rozšírenie bolo vyvíjané pre verziu 4.2 LTS, odporúčam stiahnuť práve tú, aby bola zaručená kompatibilita. Po nainštalovaní a spustení Blenderu nás uvíta hlavná obrazovka:



Obrázok 1 Hlavná obrazovka Blenderu.

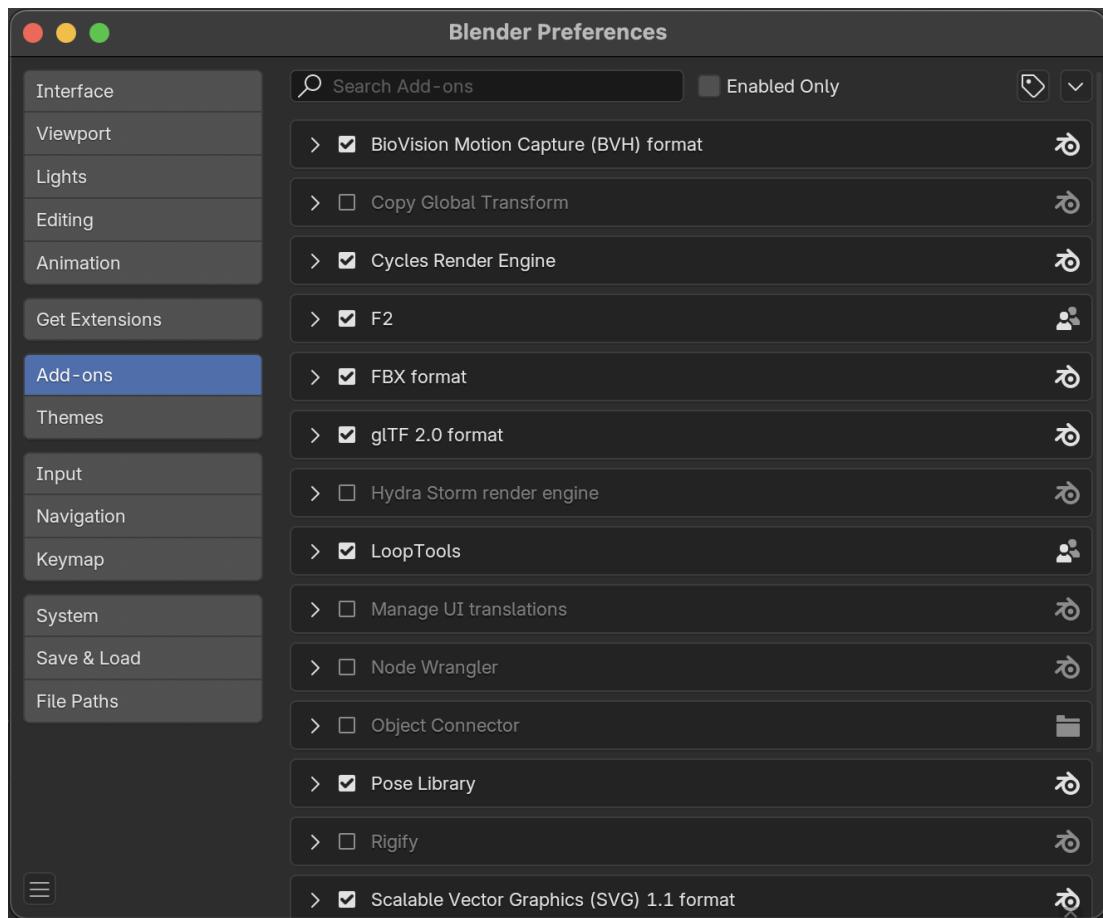
Následne je potrebné stiahnuť rozšírenie *Object Connector*, ktoré je prílohou k tejto práci. Po stiahnutí súbor *connector.py* je vhodné presunúť mimo priečinku *Downloads*. Rozšírenie teraz treba nainštalovať do Blenderu. Inštalácia rozšírení prebieha nasledovne:

1. Stlačte tlačidlo “Edit” v ľavom hornom rohu obrazovky a zobrazí sa kontextové menu.



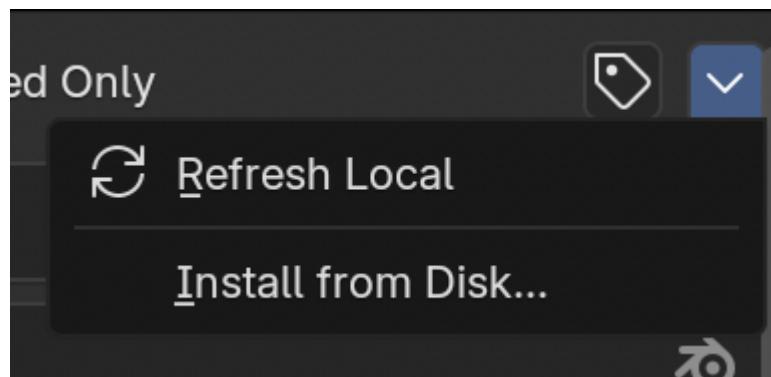
**Obrázok 2** Kontextové menu tlačidla “Edit”.

2. Po kliknutí na tlačidlo “Preferences” sa zobrazí nové okno “Blender Preferences”. V tomto okne sa nachádza kolónka “Add-ons”



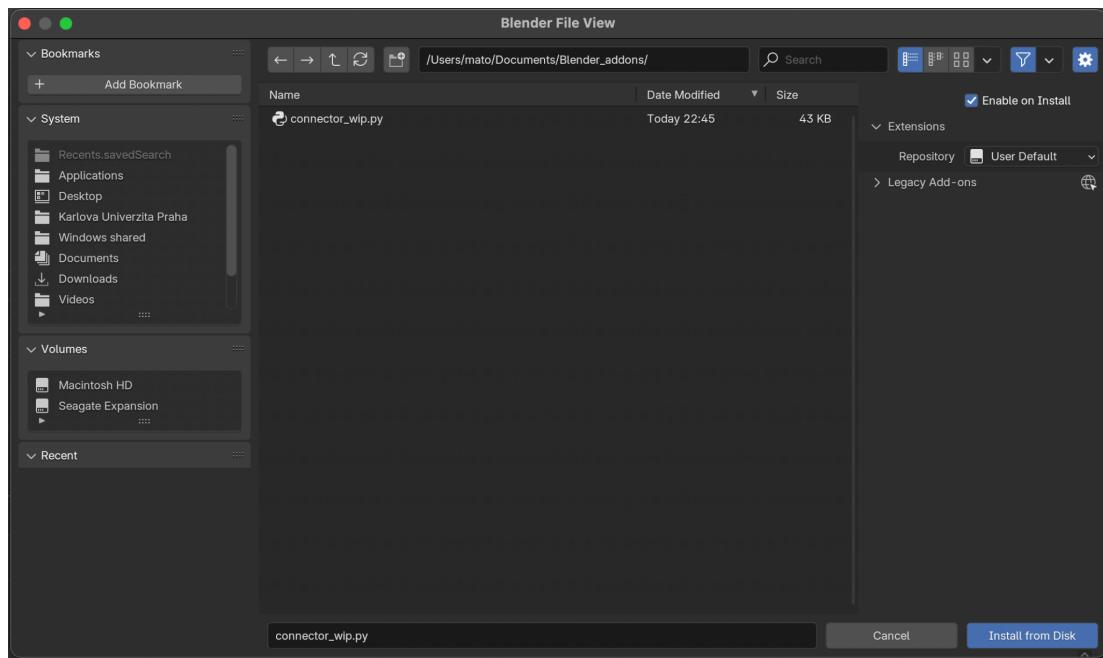
**Obrázok 3** Okno “*Blender Preferences*”.

3. V pravom hornom rohu sa nachádza šípka smerujúca nadol. Po jej stlačení sa naskytnú dve možnosti:



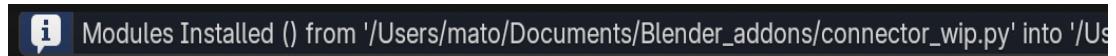
**Obrázok 4** Prídavné možnosti.

4. V tomto momente je potrebné vybrať možnosť “*Install from Disk*” a nájsť v súborovom systéme stiahnutý *connector.py*.



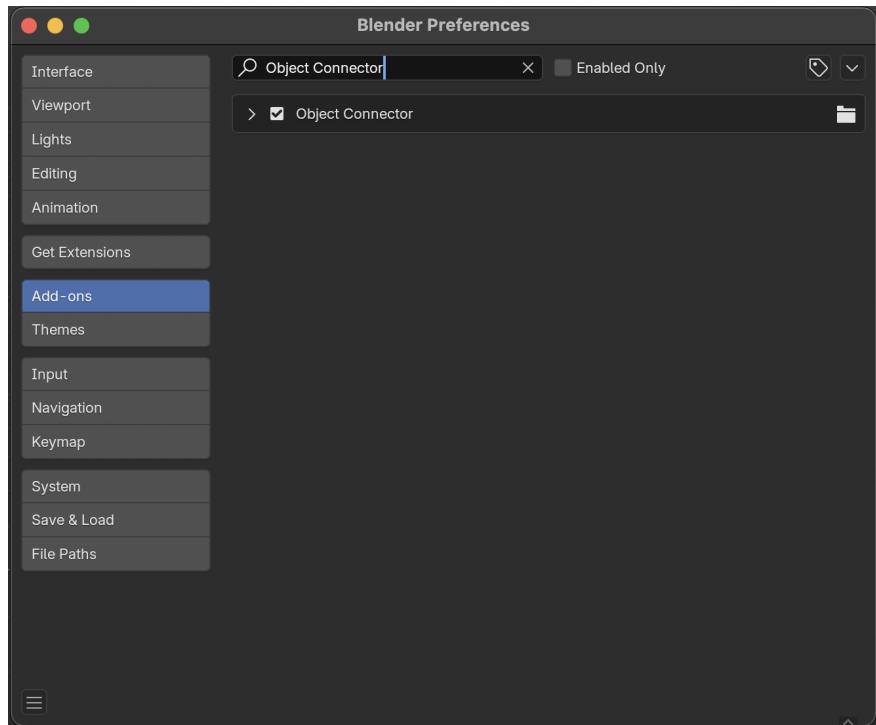
**Obrázok 5** *Blender File View.*

5. Po nájdení súboru je potrebné stlačiť tlačidlo “*Install from Disk*”. Následne sa okno “*Blender File View*” zavrie a inštaláciu potvrdí hláška v spodnej časti hlavnej obrazovky.



**Obrázok 6** Infromatívna hláška potvrdzujúca úpešnú inštaláciu.

6. Úspešná inštalácia a funkčnosť rozšírenia sa dá potvrdiť vyhľadaním “*Object Connector*” v kolónke “*Add-ons*” okna “*Blender Preferences*”.

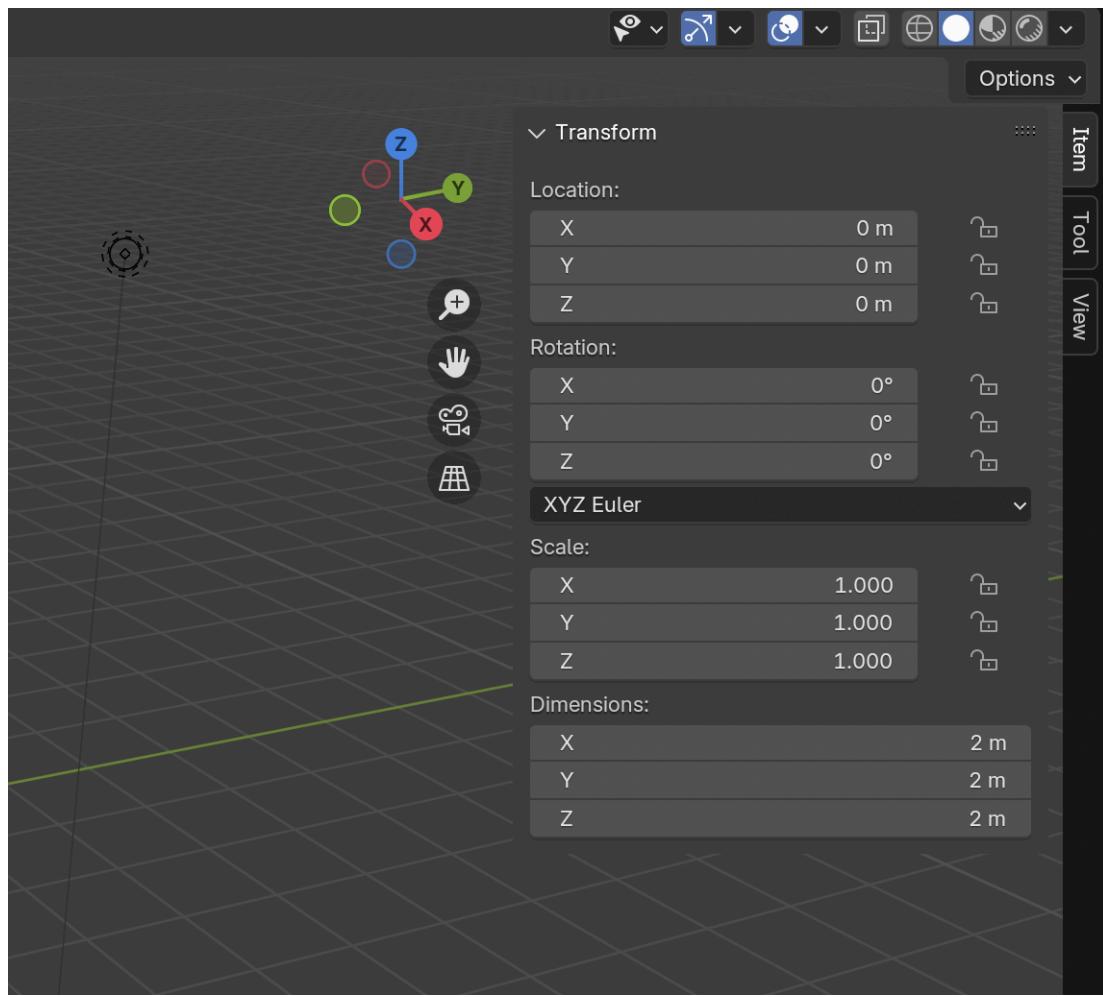


**Obrázok 7** Potvrdenie nainštalovaného a spusteného *Add-onu*.

7. V tomto momente je inštalačia dokončená a okno “*Blender Preferences*” je možné zavrieť.

## Použitie

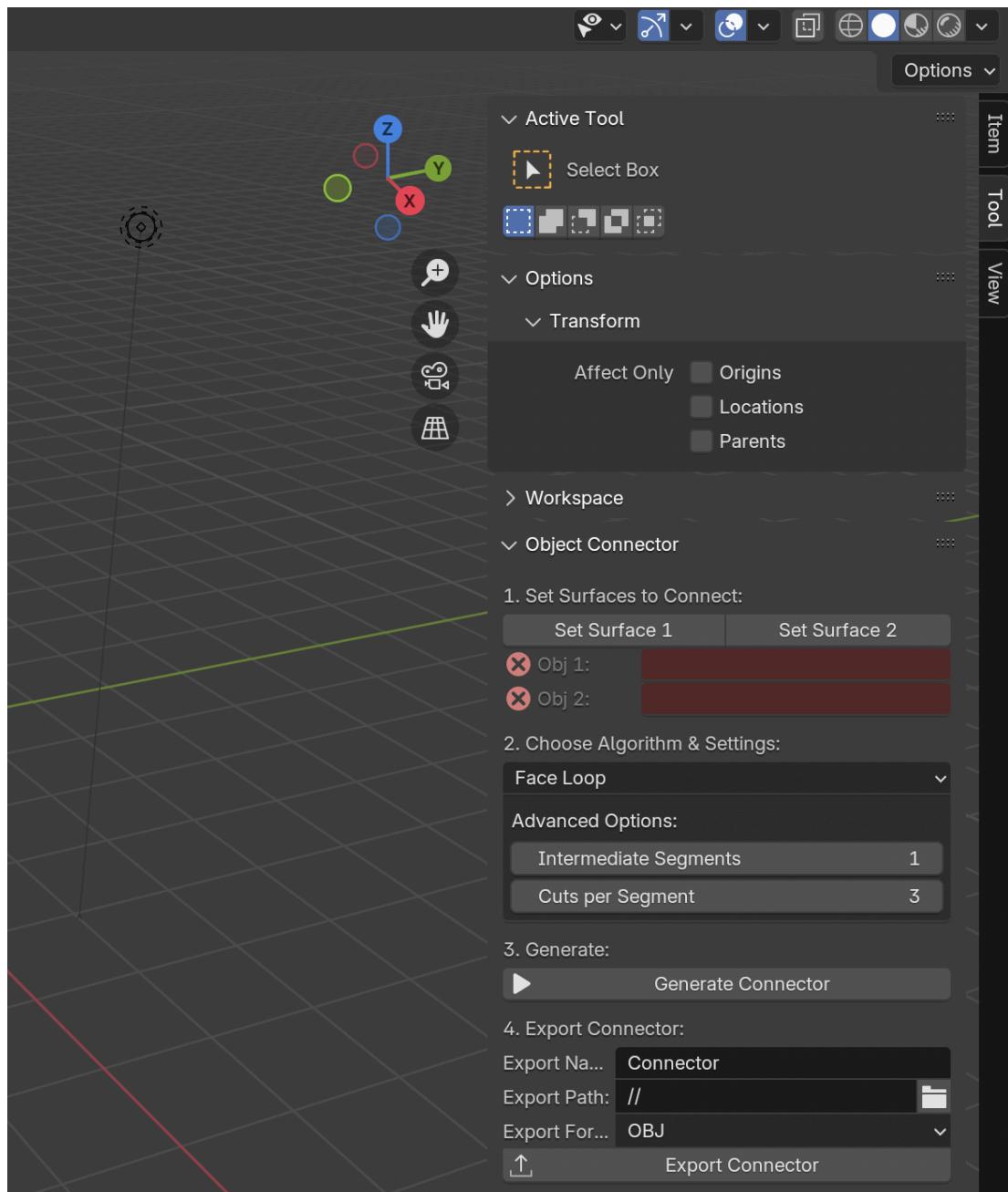
Vzhľadom na rozmanitosť scenárov, v ktorých je toto rozšírenie možné využiť, bude najjednoduchšie popísať funkčnosť na zopár príkladoch. Pre prístup k *Object Connectoru* je potrebné stlačiť tlačidlo *N*, ktoré otvorí postranný panel v pravom hornom rohu *viewportu*.



Obrázok 8

Postranný panel

Ďalším krokom je stlačenie tlačidla “*Tool*”, ktoré otvorí nasledujúce menu.



Obrázok 9 „Tool“ menu obsahujúce aj *Object Connector*.

V tomto momente už vidíme používateľské rozhranie *Add-onu*. Môžeme vidieť, že kroky sú jednoznačne logicky rozdelené a v nasledujúcej časti popíšem, ako s nimi pracovať.

## Algoritmy a kompatibilné typy povrchov

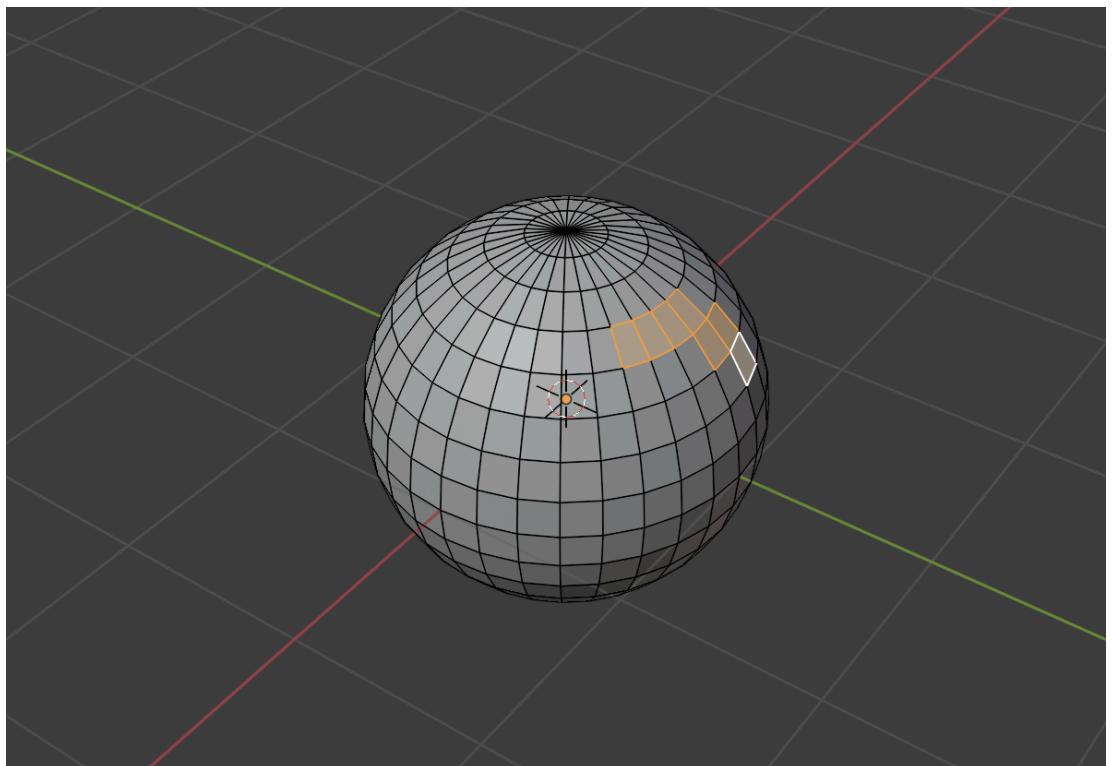
### *Simple Bridge*

Jediným platným vstupom je voľba jednej steny na každom z objektov, ktorý chceme prepojiť. Na tvari (počte hrán) tejto steny nezáleží, ale čím podobnejšie sú tvary stien, tým bude výsledok krajší.

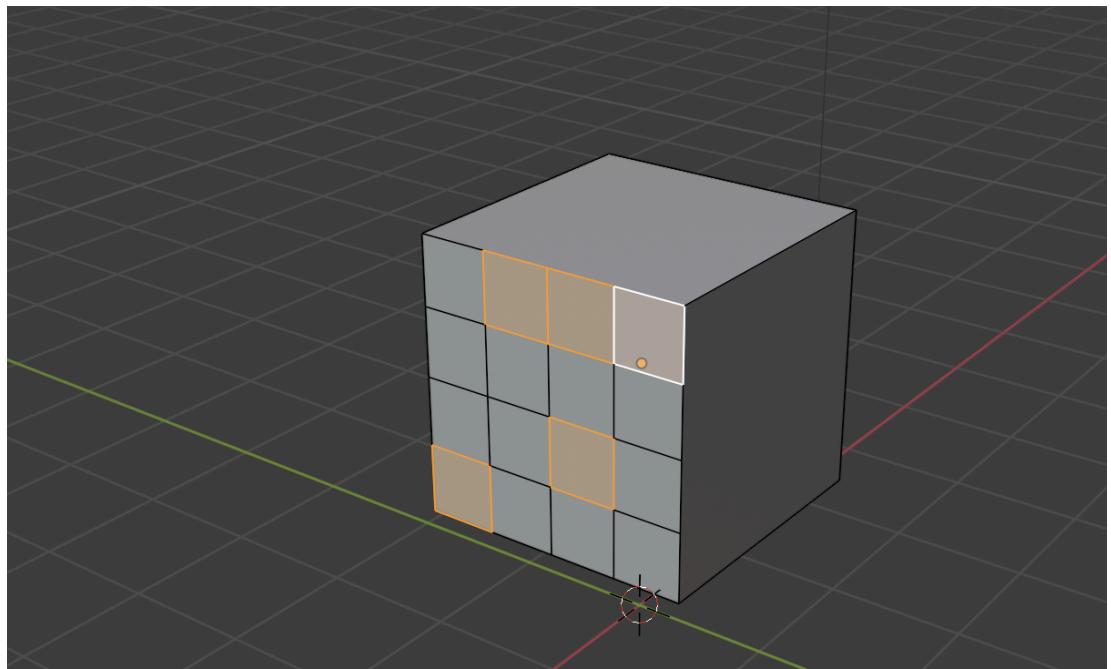
### *Closed Cap*

Ponúka viaceré možnosti platných vstupov, vrátane jednej *face* na každom objekte ako pri *Simple Bridge*. Validným vstupom je aj výber väčšieho množstva *faces* ako jedného (alebo aj oboch) z povrchov (Pre zaručenie najlepších výsledkov je dôležité, aby steny vo výbere boli čo možno najviac pri sebe.) Avšak je nevyhnutné, aby pri výbere väčšieho množstva bolo dodržané nasledovné:

- *Faces* musia byť prepojené: v praxi to znamená, že sa musí dať dostať z každej *face* do každej len prechádzaním po susedných stenách. Platným výberom teda nebude viacero “ostrovčekov”, ktoré nie sú spolu nijak spojené.

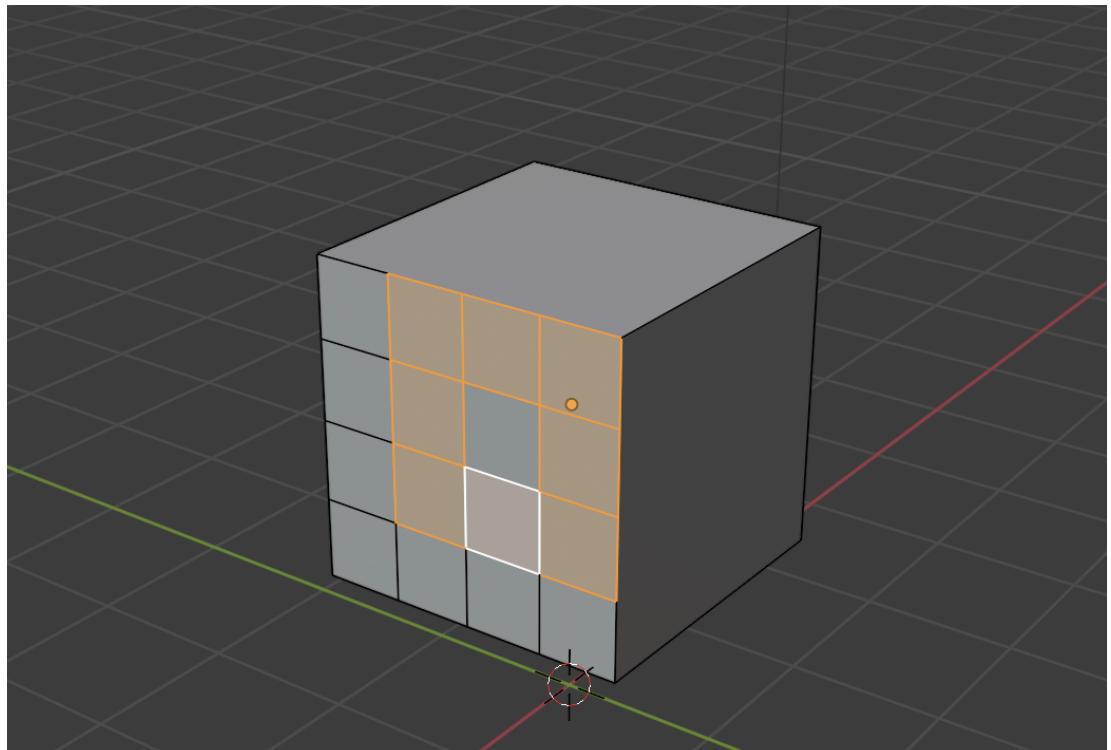


**Obrázok 10** Príklad platného vstupu pre algoritmus *Closed Cap*.



**Obrázok 11** Príklad neplatného vstupu kvôli nespojitosťi výberu

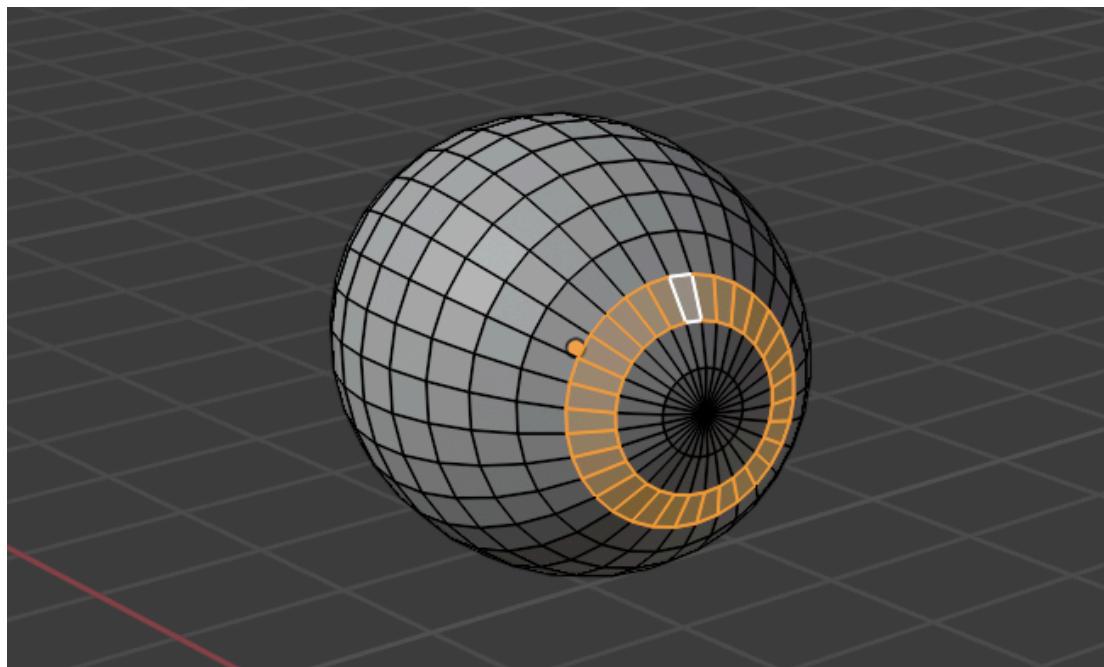
- *Faces* nemôžu obsahovať “diery”. Ak by steny tvorili “kružnicu” s nevybraným stredom, takýto vstup nebude platný.



**Obrázok 12** Príklad neplatného vstupu kvôli “diere” v strede výberu

### *Face Loop*

Ponúka len jeden typ platného vstupu. Posledný príklad neplatného výberu pre *Closed Cap* je práve vhodný vstup pre *Face Loop*. Ako vychádza z návzu, *Face Loop* je navrhnutý na spájanie takýchto “slučiek” tvorených stenami.



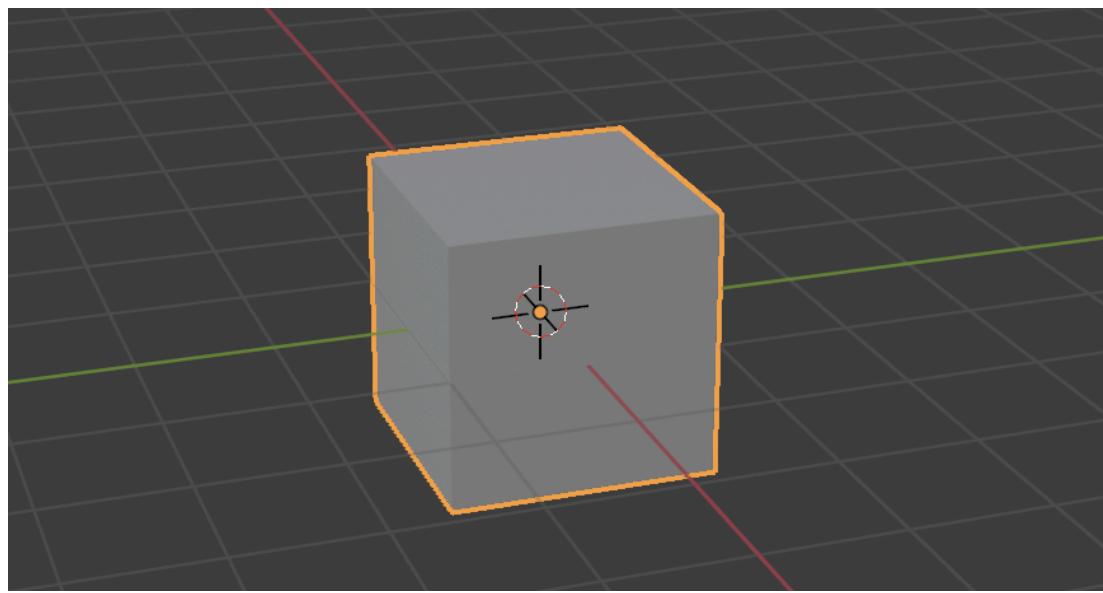
**Obrázok 13** Ďalší príklad správneho výberu povrchu pre *Face Loop*.

### Set Surfaces to Connect

Toto je krok, v ktorom je potrebné vybrať dva rôzne povrchy na spojenie. Výber povrchov je ovplyvnený algoritmom, ktorý chceme použiť.

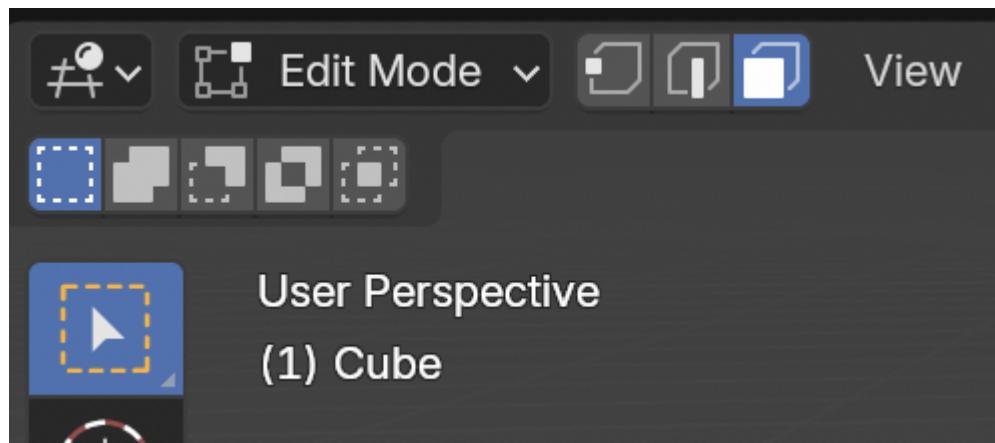
Následovne:

1. Klikneme na objekt, ktorého stenu chceme vybrať, objekt bude zvýraznený nažlto.



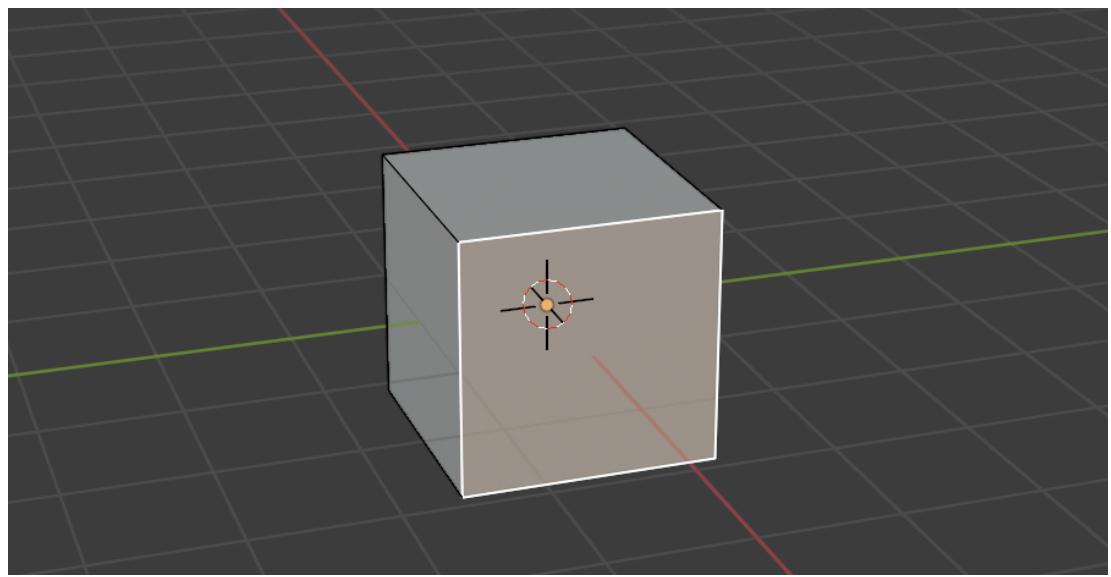
**Obrázok 14** Vybraný objekt v *Object Mode*.

2. Stlačíme *Tab*, ktoré prepne obrázok do *Edit Mode*. V ľavom hornom rohu sa zobrazí nápis *Edit Mode*, ktorý nám potvrdí, že používame správny režim.
3. Teraz je dôležité, aby bola na modro zvýraznená tretia ikonka napravo, vedľa *Edit Mode*. Toto nastavenie zaručí, že vyberáme steny (*faces*).



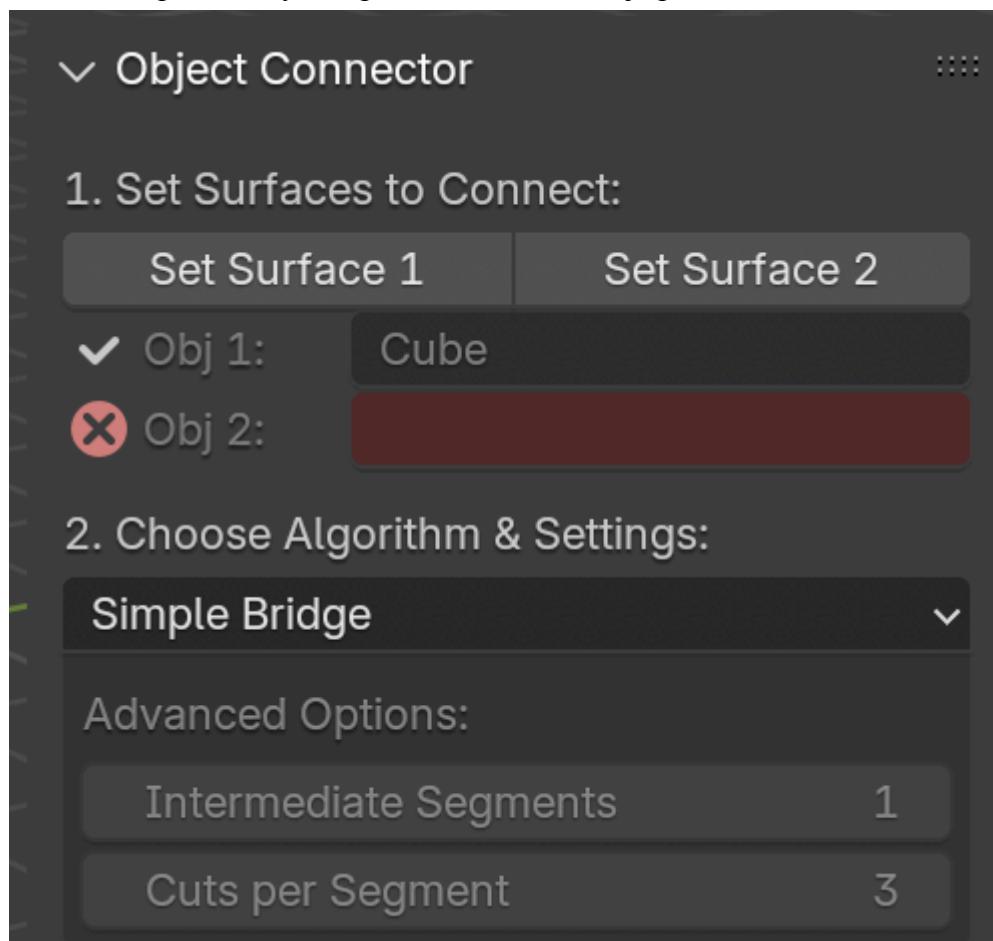
**Obrázok 15** *Edit Mode a Face Selection Mode*.

4. V tomto momente je možné vyberať *faces*. Po kliknutí na ľubovoľnú stenu nášho objektu sa stena zvýrazní nažľto. Ak by sme chceli vyberať viac *faces* ako jeden povrch (podľa návodu vyššie), stačí pridržať *Shift*, ktorý umožňuje zvoliť viacero stien naraz. Pri výbere povrchov pre algoritmus *Face Loop* sa dá vo väčšine prípadov výber zjednodušiť pridržaním tlačidla *Alt* (alebo *Option* na počítačoch *Mac*) a kliknúť na stenu, ktorú chceme. Výsledkom by mal byť výber *face loopu* obsahujúceho príslušnú stenu.



**Obrázok 16** Vybraná *face* v *Edit Mode*.

5. Následne môžeme stlačiť jedno z tlačidiel “*Set Surface 1*”, alebo “*Set Surface 2*”. O úspešnom výbere povrchu nás informuje používateľské rozhranie.

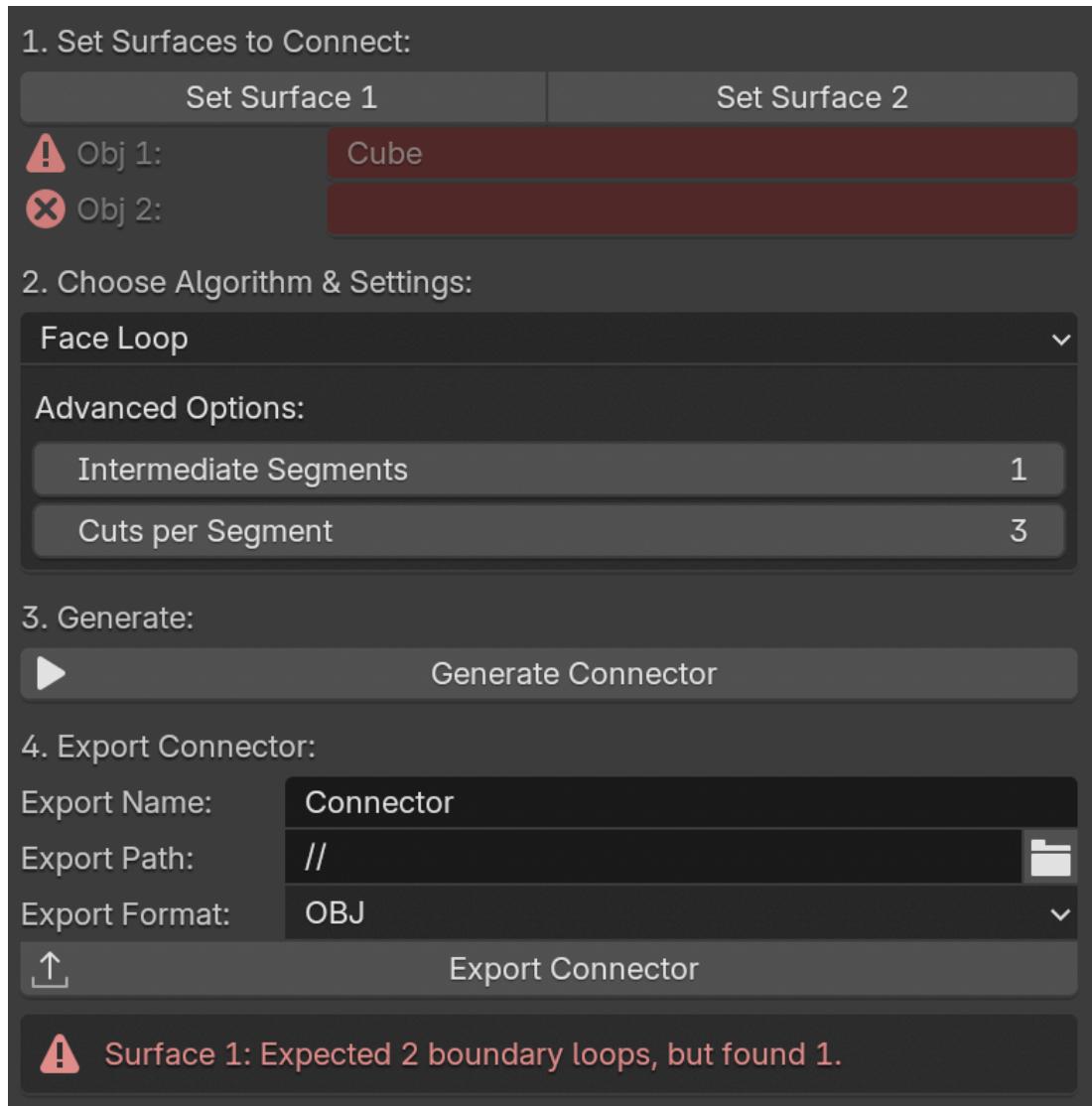


**Obrázok 17** Úspešný výber jedného z povrchov.

6. Prepneme do *Object Mode* pomocou klávesy *Tab* a kroky 1-5 zopakujeme na druhom objekte.

7. Po úspešnom výbere oboch kontaktných plôch môžeme zmeniť niektoré z *Advanced options* (popísané v ďalšej časti), alebo stlačiť tlačidlo “*Generate Connector*”.

Ak by sme teraz zvolili napríklad algoritmus *Face Loop*, pre ktorý jedna stena nie je platným povrchom na spojenie, *UI* by nám zobrazilo nasledovnú chybu:

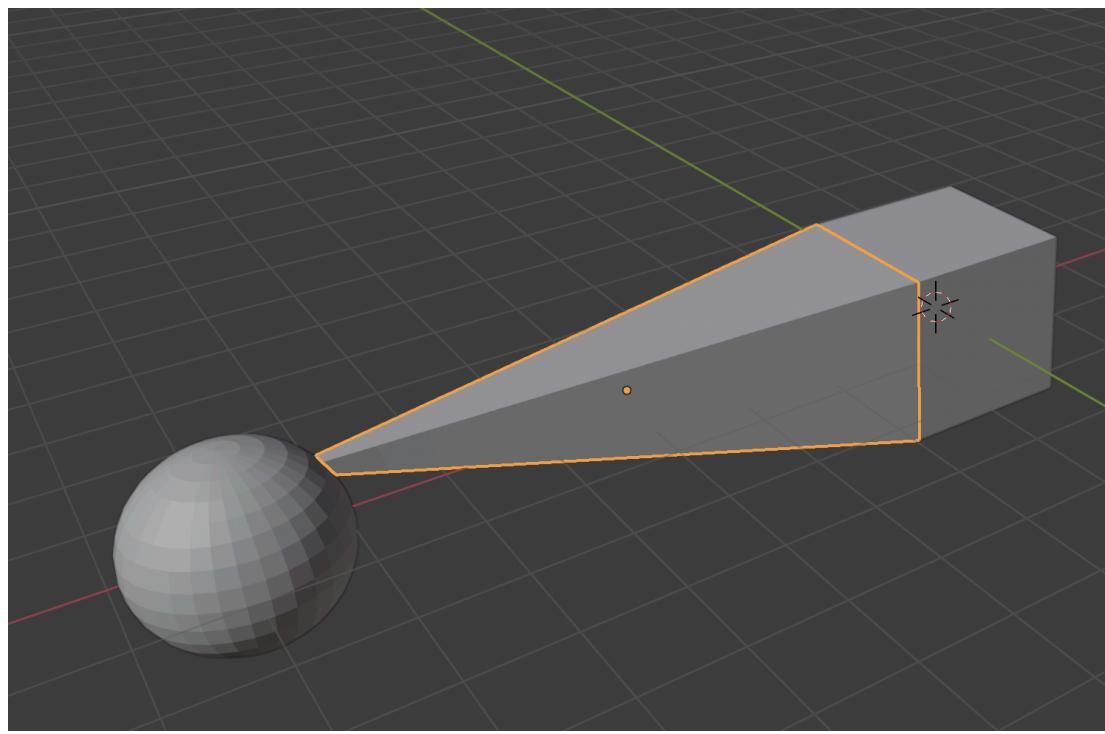


**Obrázok 18** Nesprávny výber kontaktnej plochy pre daný algoritmus.

## Prehľad dostupných algoritmov a generovanie spojovacích dielov

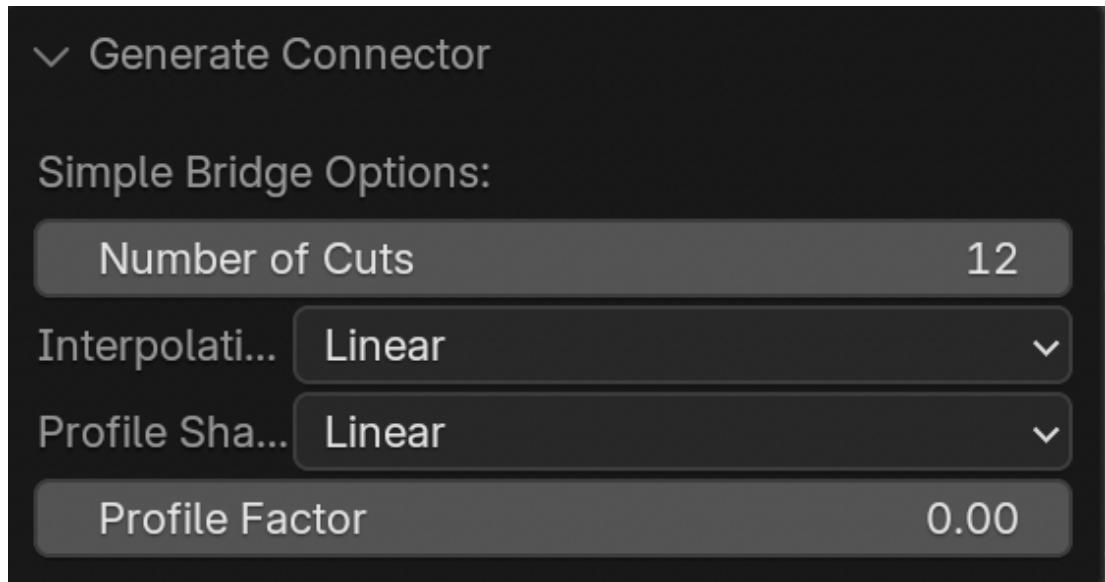
### *Simple Bridge*

Najjednoduchší algoritmus spája dve steny jednoduchým prepojením. Poskytuje *Redo Last Operation* panel, ktorý umožňuje meniť tvar medzikusu v reálnom čase.



**Obrázok 19** Medzikus vygenerovaný metódou *Simple Bridge*, bez zmeny parametrov

Po stlačení tlačidla “*Generate Connector*” sa v ľavom dolnom rohu *viewportu* zobrazí okno ponúkajúce možnosť zmeny parametrov.

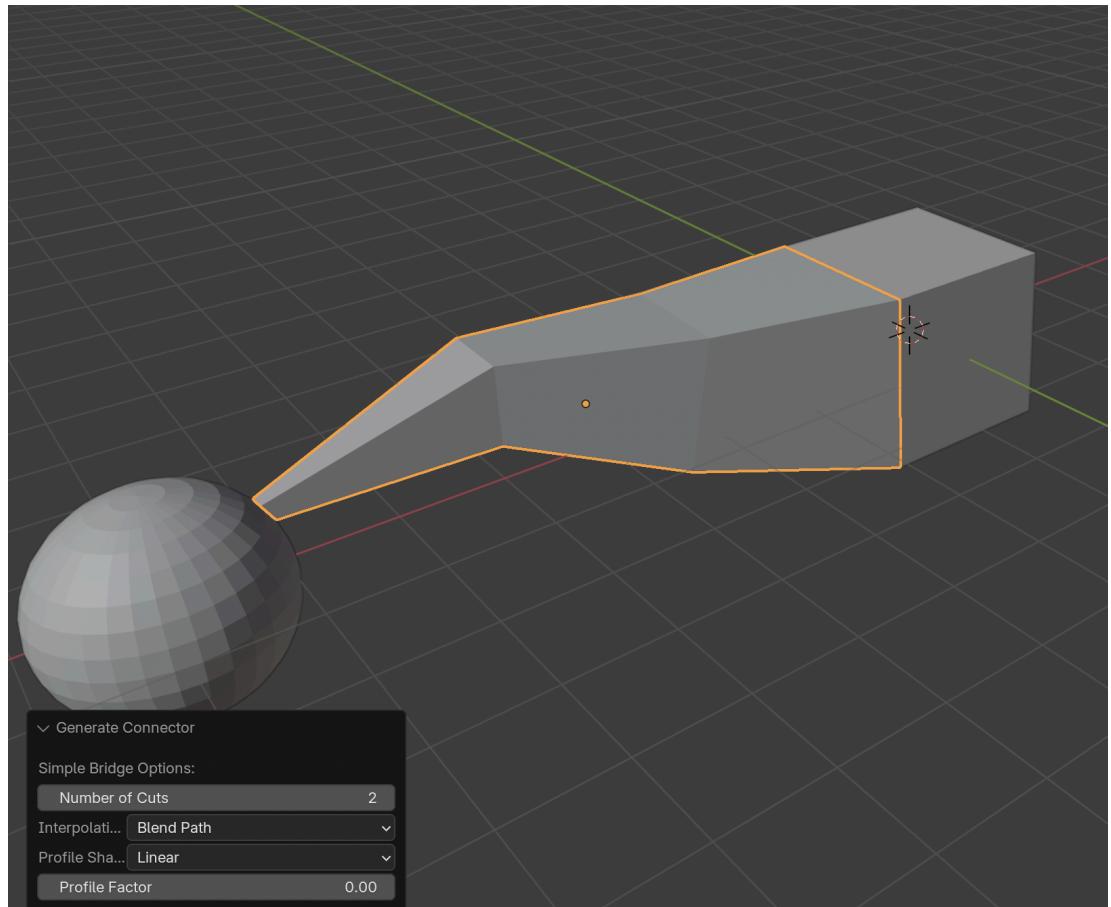


**Obrázok 20** *Simple Bridge options*

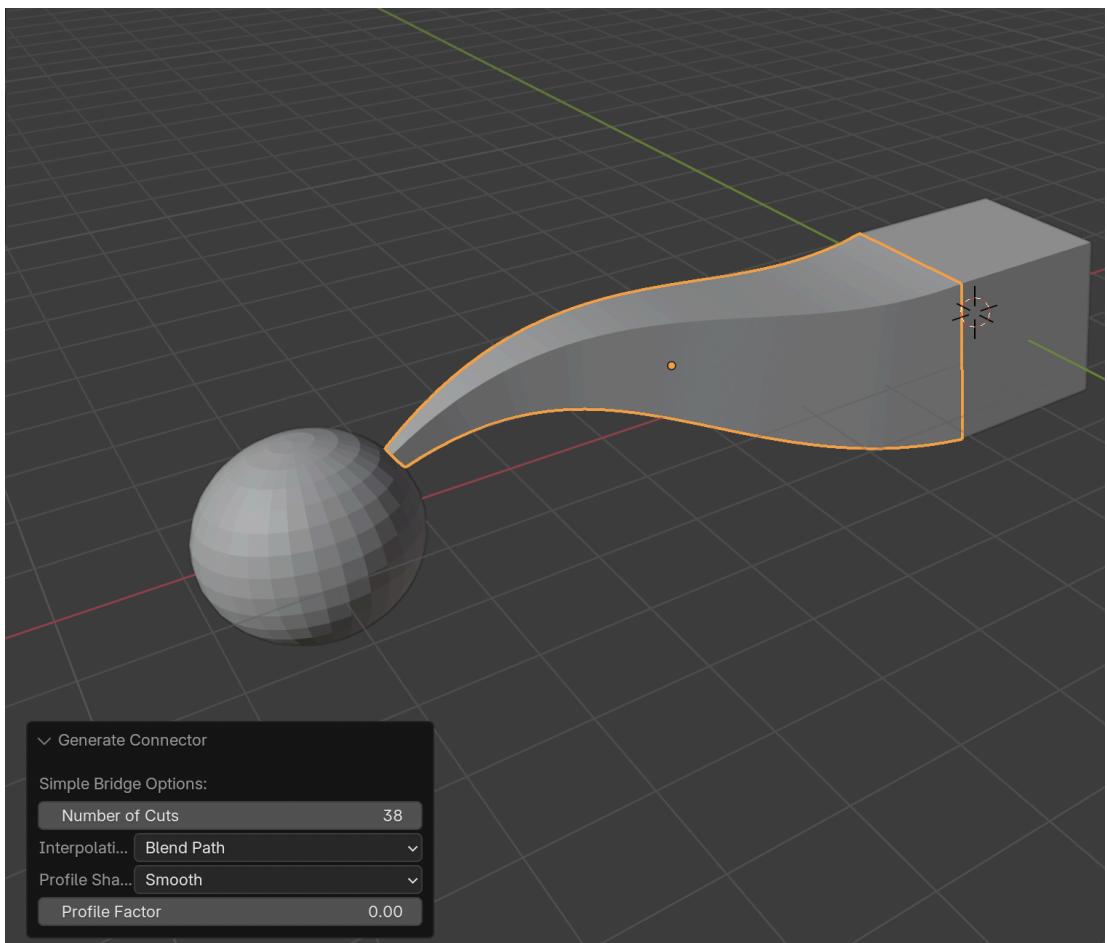
Parametre, s ktorými sa dá manipulovať:

- *Number of Cuts*: počet segmentov, na ktoré bude prepojovací diel rozdelený. Čím bude táto hodnota vyššia, tým plynulejší bude medzikus. Vizuálna zmena ale nastane iba v prípade, že zmeníme interpoláciu z lineárnej na *Blend Path*.

- *Interpolation*: výber interpolácie, ktorú chceme, aby algoritmus použil.  
Lineárna vráti najkratšie, “ostré” spojenie, zatiaľ čo *Blend Path* vráti organickejší tvar.



**Obrázok 21** Príklad *Blend Path* interpolácie s nízkym počtom rezov.



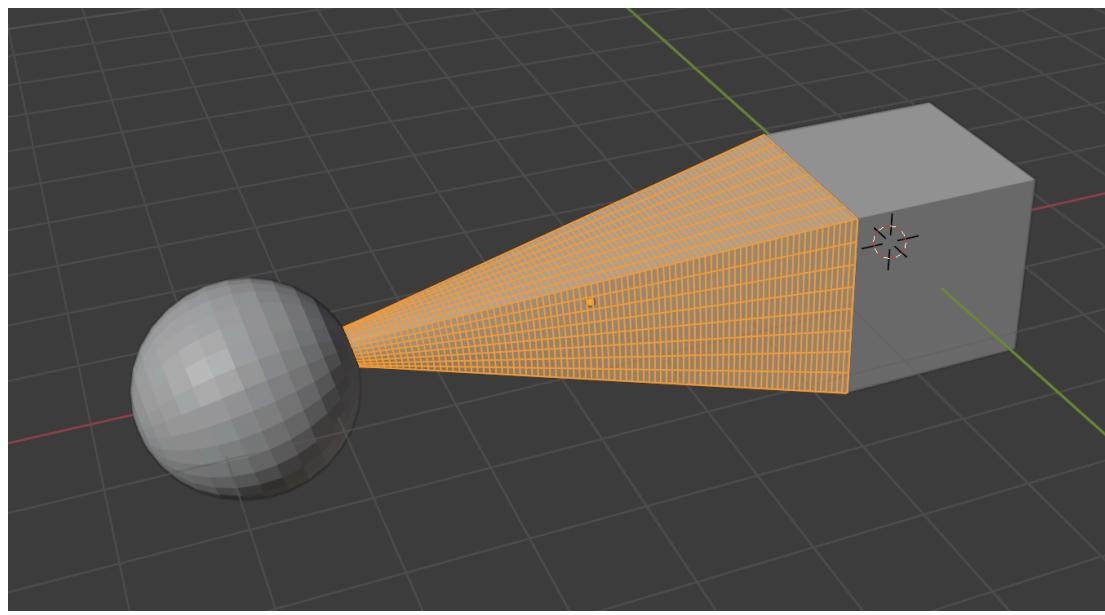
**Obrázok 22** Príklad *Blend Path* interpolácie s vyšším počtom rezov.

- *Profile shape*: Tento parameter ovláda bočný profil alebo “zakrivenie” premostenia medzi dvoma kontaktnými plochami. Má zopár možnosti, ktoré sa dajú zvoliť. Tieto možnosti budú demonštrované na obrázkoch.
- *Profile factor*: určuje, ako veľmi sa zvolený tvar profilu prejaví. Pri hodnote 0.0 je efekt úplne vypnutý, pri hodnote vyššej ako 0.0 sa medzikus “nafúkne” podľa tvaru profilu. Pri záporných hodnotách sa efekt obráti.

Najlepší spôsob, ako s týmito parametrami pracovať je najprv zvoliť požadovaný tvar profilu a ten následne pomocou *Profile Factor* zväčšovať Alebo zmenšovať podľa potreby.

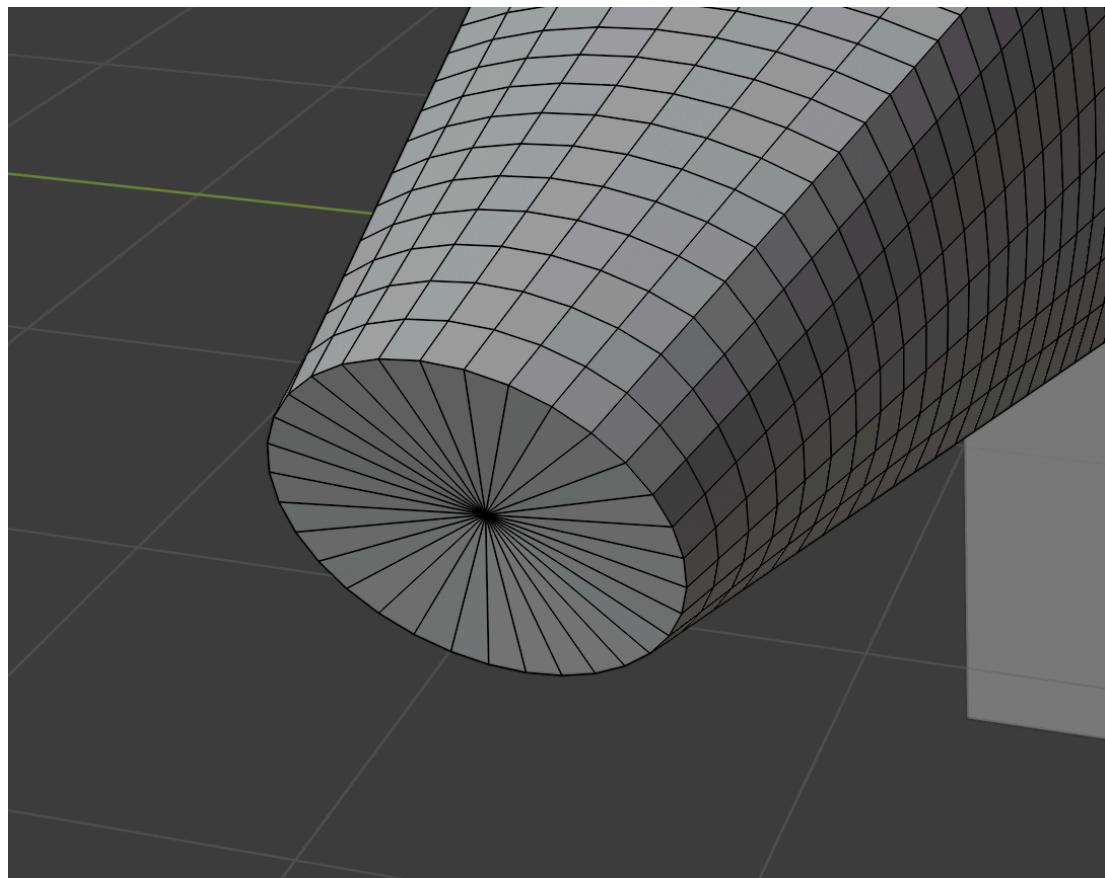
### *Closed Cap*

Ide o pokročilejší algoritmus, ktorý funguje na inom princípe ako *Simple Bridge*. Pri tomto algoritme máme k dispozícii zmeny počtu segmentov a rezov v nich. Tieto zmeny musia byť spravené pred generovaním samotného prepojovacieho dielu.



**Obrázok 23** Príklad prepojenia vrchu gule s jednou stenou kocky. *Number of segments = 20, Cuts per segment = 3*

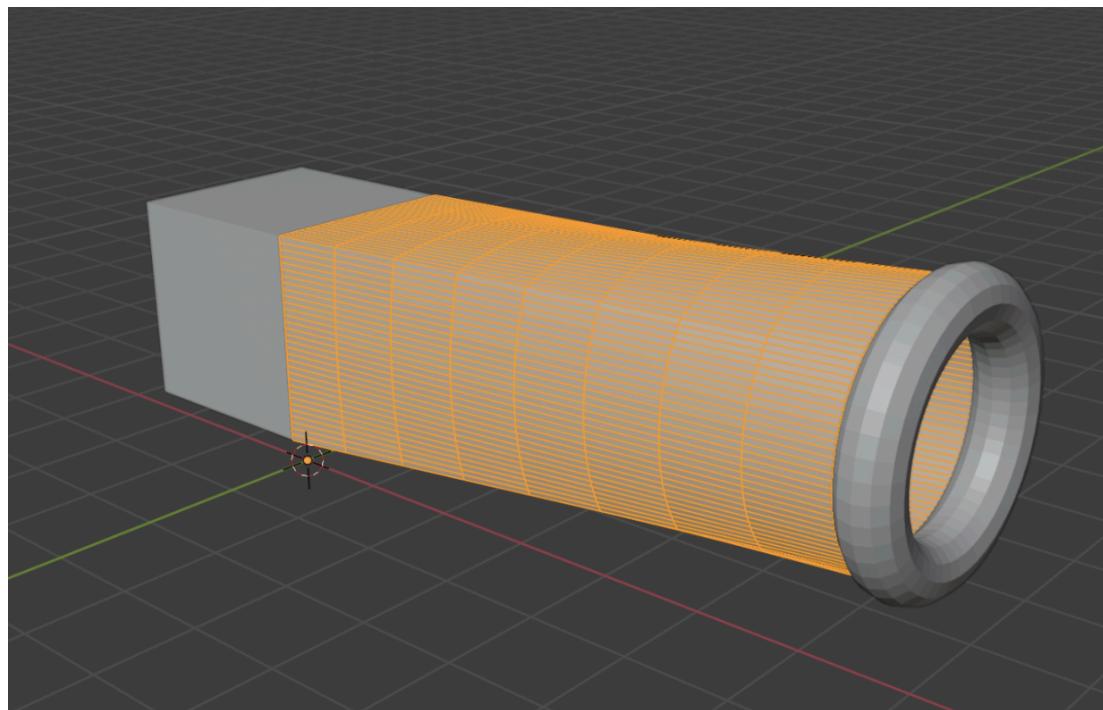
Zmenou nastaviteľných parametrov je možné docieliť hladkejší povrch, hlavne ak je aspoň jeden z prepojovaných povrchov guľatý.



**Obrázok 24** Ukážka hladkosti medzikusu z predošlého príkladu

## *Face Loop*

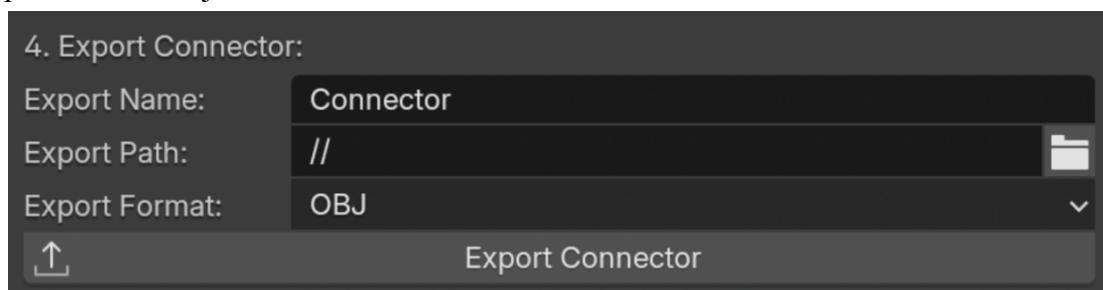
Najpokročilejší typ algoritmu, ktorý je vhodný na prepájanie objektov s dvojvrstvovou štruktúrou stien (napríklad potrubia). Ponúka rovnaké možnosti zmeny úrovne detailu ako *Closed Cap*.



**Obrázok 25** Príklad medzikusu vygenerovaného pomocou algoritmu *Face Loop*.

## Export

Po úspešnom vygenerovaní prepojovacieho dielu bude vybraný a môžeme s ním voľne manipulovať (stlačením klávesy “G” a pohybom myšou). Ak sme spokojní s výsledkom, môžeme pokračovať exportom do vhodného formátu. Export panel ponúka nasledujúce možnosti:



**Obrázok 26** Export panel.

- *Export name*: meno, ktoré chceme použiť pre nás výsledný súbor. V prípade, že žiadne meno nebude zvolené, použije sa to, ktoré má objekt vo *viewporte*.
- *Export path*: Cesta v súborovom systéme. Predvolená cesta “//” exportuje medzikus do rovnakého priečinka, v akom sa nachádza aktuálny *.blend* súbor. V prípade, že pracujeme v neuloženom *.blend* súbore, chybová hláška nás upozorní na nemožnosť exportu do tejto lokality. Cestu vieme napísat ručne

do polička, alebo stlačením ikony sa dostaneme do prehliadača súborov, kde môžeme požadovaný priečinok na export nájsť.

- *Export format*: Na výber sú dva najčastejšie používané formáty pre 3D objekty alebo 3D tlač.

Ked' sme spokojní s našim výberom, stlačíme tlačidlo “*Export Connector*” a v spodnej časti obrazovky sa zobrazí správa, informujúca nás o úspešnom exporte.



Obrázok 27 Potvrdenie exportu.

## Programátorská dokumentácia

Tento projekt je vyvinutý čisto v jazyku Python, konkrétnie vo verzii 3.11. Používa štandardné knižnice dostupné len vo vnútornom prostredí Blenderu (*bpy*, *bmesh*). Pre uľahčenie vývoja som pracoval s rozšíreniami ako je *fake-bpy-module* od *nuttı* [<https://github.com/nuttı/fake-bpy-module>] a *Blender Development Tools* od *Jacquesa Luckeho* [[https://github.com/JacquesLucke/blender\\_vscode?tab=readme-ov-file](https://github.com/JacquesLucke/blender_vscode?tab=readme-ov-file)]. Celá funkciuálnita je obsiahnutá v jedinom .py súbore ([connector.py](#))

### Štruktúra kódu

Kód je rozdelený do niekoľkých logických celkov:

\paragraph{Vlastnosti a dáta}

Stav programu a používateľské vstupy sú uložené ako vlastnosti priamo do Blender scény (*bpy.types.Scene*), čo zaručuje globálnu prístupnosť dát pre všetky časti kódu.

- *FaceIndexPropertyGroup*: Vlastná dátová štruktúra slúžiaca ako *container* pre celé číslo (*IntProperty*). Používa sa v *CollectionProperty* na ukladanie indexov *faces*.
- *connector\_surfaces\_1* a *connector\_surfaces\_2*: dve *CollectionProperties* typu *FaceIndexPropertyGroup*, uchovávajú zoznamy indexov stien pre oba povrchy (ktoré spájame)
- *connector\_obj\_1*, *connector\_obj\_2*: mená prepájaných objektov typu *StringProperty*.
- *connector\_status* (1 a 2), *connector\_error* (1 a 2): vlastnosti na ukladanie stavu validácie, ktoré *UI* číta na zobrazenie vizuálnej spätej väzby (chybové hlášky, zmena ikon)
- *connector\_topology\_type*: *EnumProperty*, ktorej možnosti sú *FACE\_LOOP*, *CLOSED\_CAP* a *SIMPLE\_BRIDGE*. Tieto možnosti určujú, ktorý algoritmus bude použitý a zmena tejto *EnumProperty* spôsobí zavolanie

*topology\_type\_update callbacku*. Jeho zavolanie spôsobí spustenie funkcie *run\_validation()*, ktorá overí, či vstupy sú platné pre novozvolenú možnosť v *connector\_topology\_type*.

- *connector\_number\_segments* a *connector\_number\_cuts*: nastavenia typu (*IntProperty*) pre pokročilé algoritmy.
- *connector\_obj*: *StringProperty* reprezentujúca meno vygenerovaného medzikusu.
- *export\_...* : vlastnosti (*EnumProperty*, *StringProperty*) pre nastavenie exportu (formát, cesta, názov súboru)

#### \paragraph{Operátory}

tri hlavné triedy, ktoré reagujú na interakciu používateľa.

- *OBJECT\_OT\_SelectConnectorSurface*: spustí sa po stlačení tlačidla “*Set Surface* (1 alebo 2)” a jeho úlohou je identifikovať aktuálny vybrané povrhy v *Edit Mode*, získať ich indexy a uložiť ich spolu s menom objektu do príslušných vlastností v scéne. Potom zavolá *run\_validation()*.
- *OBJECT\_OT\_GenerateConnector*: Hlavný komponent tohto rozšírenia, ktorý sa spustí po kliknutí na “*Generate Connector*”. Načíta uložené dátá a overí, či oba vybrané povrhy majú status “*VALID*”. Ak majú, na základe zvoleného algoritmu (*connector\_topology\_type*) zavolá príslušnú *execute\_* metódu. Obsahuje tiež logiku na vytváranie dočasných “koncoviek” (*end caps*), aby bol na konci medzikus “vodotesný”, ktoré na konci k modelu pripojí.
- *OBJECT\_OT\_ExportConnector*: jednoduchý operátor, ktorý načíta meno vygenerovaného objektu zo scény (v prípade, že by používateľ nezvolil vlastné) a uloží objekt do súboru vo zvolenom formáte (*EnumProperty* - .obj alebo .stl).

#### \paragraph{UI}

- *OBJECT\_PT\_ObjectConnectorPanel*: definite panel kolónke *Tools* vo *viewporte*. Táto trieda je zodpovedná za vykreslenie všetkých vizuálnych prvkov. Taktiež číta *connector\_status* vlastnosti zo scény, na základe ktorých mení vzhľad *UI* (napr. Zobrazenie chybových hlášok).

#### \paragraph{Validačná logika}

Validácia vstupu v reálnom čase dáva používateľom okamžitú spätnú väzbu o vhodnosti ich výberu.

- *validate\_selection\_util(scene, surface\_index)*: Kontroluje, či výber daného povrchu existuje, či je platný (či má správny počet *boundary loops*) a či neobsahuje nesúvislé skupiny *faces*.
- *run\_validation(context)*: funkcia, ktorá je volaná po každej zmene výberu povrchov. Zavolá *validate\_selection\_util* pre oba povrhy a vykoná testy medzi nimi (napr. či sa neprekryvajú, alebo či je o ten istý povrch). Výsledky (status tejto kontroly a prípadnú chybovú hlášku) uloží do vlastností (*connector\_status*, *connector\_error*), odkiaľ ich *UI* prečíta.

## \paragraph{Pomocné funkcie}

Vykonávajú dôležité výpočty, sú volané z operátorov.

- *duplicate\_face\_loops()*: duplikuje vybrané plochy, aby sa dali neskôr použiť na “zacelenie” medzikusu.
- *get\_boundary\_loops\_data()*: Analyzuje geometriu a nájde všetky otvorené hrany (*boundary edges*), z ktorých poskladá usporiadane *edge loops*. Ich súradnice prepočíta do globálneho priestoru maticovým násobením a zmeria ich dĺžku
- *resample\_loop\_coords()*: Základná funkcia pokročilých algoritmov. Umožňuje nedeštruktívne vyrovnať počet vrcholov prepájaných plôch.
- *align\_loops\_by\_cost()*: Zabráňuje *twistingu* medzikusu tým, že nájde optimálnu rotáciu jedného *loopu* voči druhému.

## \paragraph{Registrácia}

Funkcie *register()* a *unregister()* nasledujú štandardy vývoja rozšírení pre Blender.

*register()* registruje všetky triedy a definuje všetky potrebné vlastnosti v scéne.

Funkcia *unregister()* vykoná opačný proces, aby po deaktivácii *add-onu* nezostali v Blenderi žiadne pozostatky.

## Možnosti rozšírenia

- RefaktORIZÁCIA kódu: komplexné funkcie by sa dali dekomponovať na menšie funkcie pre lepšiu čitateľnosť a údržbu.
- Nové algoritmy generovania medzikusu: nové algoritmy sa dajú pridať jednoducho rozšírením *connector\_topology\_type* a navrhnutím príslušnej funkcie.
- Viac možností interpolácie pri *Advanced Loft* algoritnoch by bolo možné pridať nové funkcie namiesto existujúcej lineárnej interpolácie.
- Lepšie zúžitkovanie *Cuts per Segment*: tento parameter momentálne nepredstavuje funkčný rozdiel oproti *Number of Segments*. V budúcnosti by bolo možné ho zmysluplniešie zakomponovať.