České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačové grafiky a interakce

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Adam Samec**

Studijní program: Softwarové technologie a management
Obor: Web a multimedia

Název tématu: **Klient-server aplikace pro ilustrativní explodování 3D modelů**

Pokyny pro vypracování:

Seznamte se s aplikací pro ilustrativní explodování modelu a upravte aplikaci do podoby webové služby běžící na serveru. Webová služba bude schopna uložit nahraný 3D model, převést ho do reprezentace vhodné pro klient-server aplikaci (např. JSON) a vypočítat pro něj graf exploze. Dále implementujte klientskou část aplikace pomocí technologie WebGL. Klient umožní prezentovat uživateli 3D model uložený na serveru a explodovat 3D model na základě informací získaných z webové služby a regionu zájmu (jeden 3D objekt modelu) zadaného uživatelem. Funkčnost implementace ověřte alespoň na deseti 3D modelech skládajících se minimálně z deseti různých 3D objektů.

Seznam odborné literatury:

W. Li, M. Agrawala, B. Curless, and D. Salesin. Automated generation of interactive 3d exploded view diagrams. ACM Transactions on Graphics, 27(3):101:1–101:7, Aug. 2008.

Jiří Uher. Ilustrativní explodování 3D modelů. Diplomová práce, ČVUT v Praze, 2013.

Vedoucí: Ing. Ladislav Čmolík, Ph.D.

Platnost zadání: do konce letního semestru 2015/2016

[redacted]

prof. Ing. Jiří Žára, CSc.
vedoucí katedry

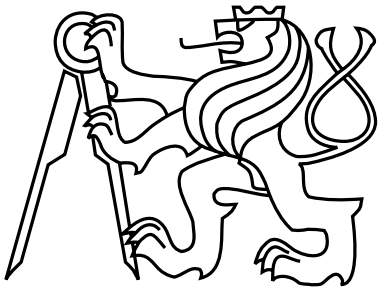
L.S.

[redacted]

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 18. 4. 2015

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačové grafiky a interakce



Bakalářská práce

# Klient-server aplikace pro ilustrativní explodování 3D modelů

*Adam Samec*

Vedoucí práce: Ing. Ladislav Čmolík, Ph.D.

Studijní program: Softwarové technologie a management, bakalářský

Obor: Web a multimedia

15. května 2015

# Poděkování

Děkuji vedoucímu práce Ing. Ladislavu Čmolíkovi za odbornou pomoc a vstřícnost. V neposlední řadě děkuji rodině a blízkým přátelům za průběžnou podporu.

# Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změněně, některých zákonů (autorský zákon).

V Praze dne 20. května 2015 .....

# Abstract

This bachelor project is concerned with modification of an existing desktop application for exploded views of 3D models into a client–server web application. The server side will enable storage of the models uploaded by the client, explosion graph computation and conversion of the models to an appropriate format for view in a web browser using the WebGL technology. The client side will provide administration, interactive view and exploding of the models based on information received from the server and a selected object as the centre of user attention.

# Abstrakt

Tato bakalářská práce se zabývá úpravou existující desktopové aplikace pro ilustrativní explodování 3D modelu do podoby webové klient–server aplikace. Serverová část bude sloužit k ukládání klientem nahraných modelů, výpočtu grafu exploze a převedení modelů do formátu vhodného pro zobrazení klientem ve webovém prohlížeči pomocí technologie WebGL. Klientská část umožní správu, interaktivní prohlížení a explozi modelů na základě informací získaných ze serverové části a uživatelem zvoleného dílčího objektu modelu jako středu zájmu.

# Obsah

# 1 Úvod

## 1.1 Význam ilustrace

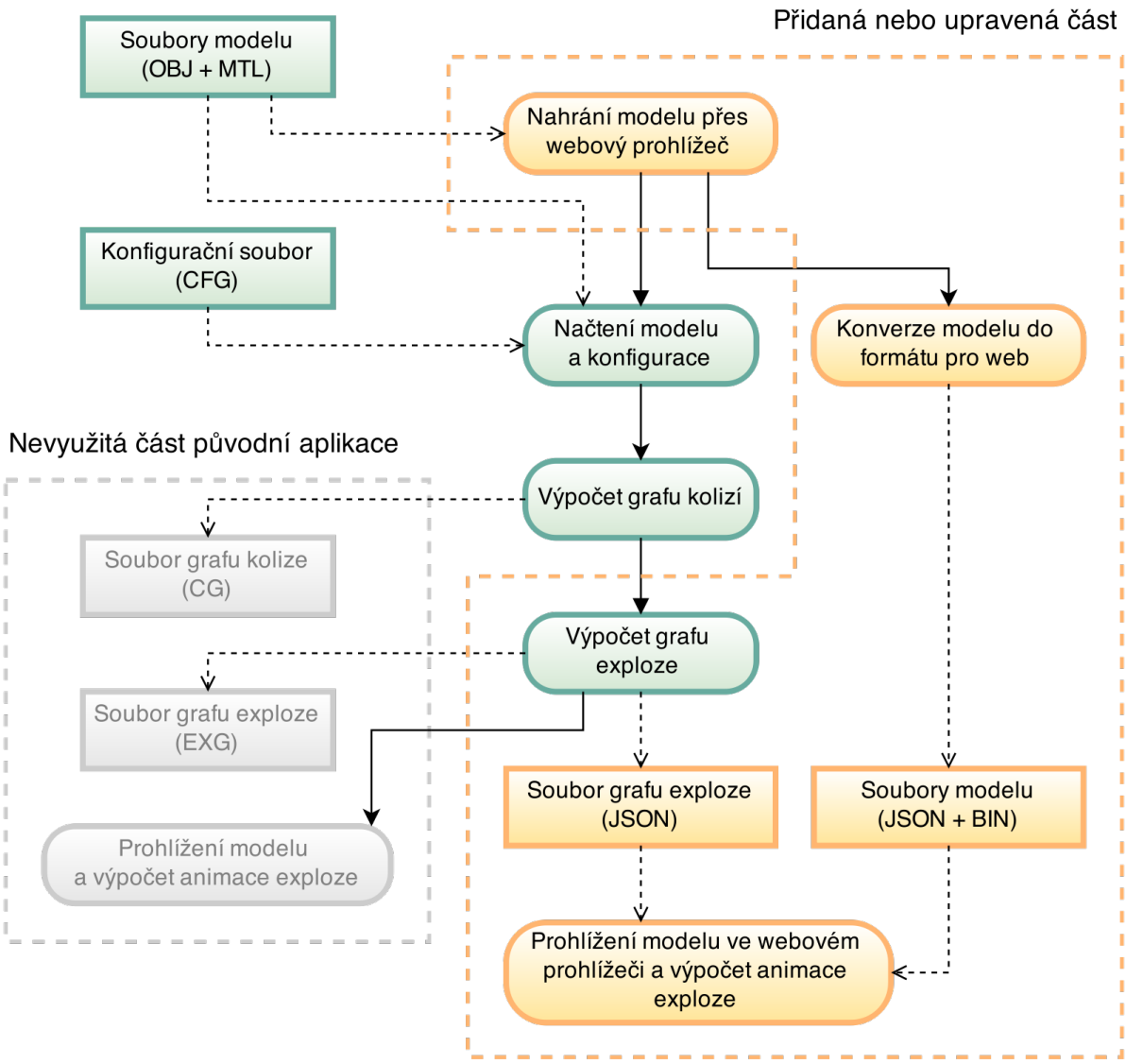
Již od pravěkých časů lidé používají ilustraci jako prostředek pro sdělování myšlenek, pocitů a rozvíjení své tvořivosti. V technice se ilustrace uplatňuje jako obrazová forma dokumentace sloužící k preciznímu popisu předmětů či vztahů mezi nimi.

Ilustrativní explodování je dalším krokem ve vývoji ilustrace, který umožňuje blíže odkrýt vnitřní strukturu znázorňovaných předmětů, v tomto případě reprezentovaných trojrozměrnými modely. Explodováním se zde rozumí animované rozložení zobrazeného modelu tak, aby byl plně viditelný jeho zvolený díl, který je středem zájmu. Postup, jakým je model rozkládán, je v našem případě vyhodnocen automaticky. Je vyžadována pouze znalost geometrie a pozice jednotlivých dílů, definice možných směrů jejich odsunutí a parametrů kompenzujících nepřesnosti v geometrii modelu.

## 1.2 Cíl práce

Tato bakalářská práce navazuje na diplomovou práci nazvanou „Ilustrativní explodování 3D modelů“ [1], jejímž autorem je Jiří Uher, bývalý student ČVUT FEL. Původní práce blíže rozebírá problematiku ilustrace technikou explodování a zkoumá existující řešení. Jejím výsledkem byla desktopová Java aplikace umožňující ilustrované explodování 3D modelů. V závěru původní práce je nastíněna možnost řešení jejích nedostatků spočívajících v procházení dílů modelu skrz sebe během explodování a v nezohledňování pohledu kamery při odkrývání zvoleného dílu.

Cíl této práce se skládá ze dvou částí. První část se zabývá analýzou původní desktopové aplikace, nalezením příčin zmíněných nedostatků a návrhem jejich řešení. Druhá část si klade za cíl úpravu desktopové aplikace do podoby webové klient-server aplikace rozšířené o jednoduchou administraci modelových souborů. Specifikem této nástavby je využití technologie WebGL pro implementaci explodovaného prohlížení modelů ve webovém prohlížeči. Webová aplikace má rovněž implementovat navržené řešení nedostatků původní aplikace. ***Obrázek 1*** ilustruje vstupy a výstupy původní aplikace, její základní funkční celky a jejich využití nově přidanou částí webové aplikace.



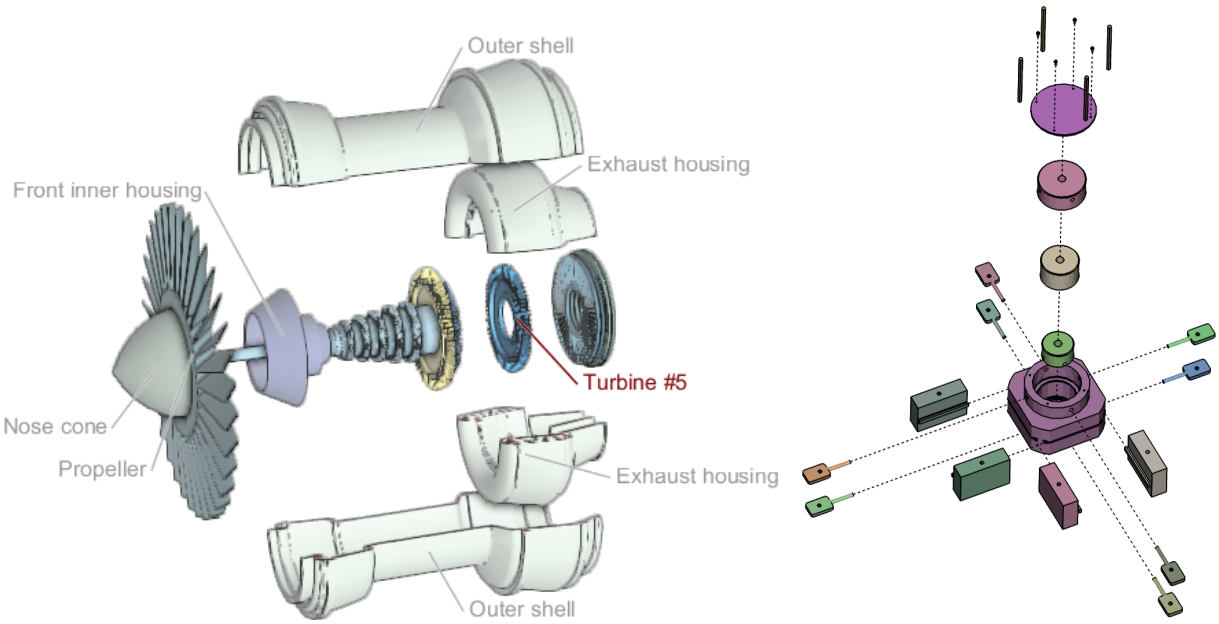

**Obrázek 1** – *Schéma změn v aplikaci.*

## 1.3 Použité technologie

WebGL je otevřený licenčními poplatky nezatížený standard pro nízkoúrovňové 3D aplikační programové rozhraní (API). Standard je navržen a spravován konsorciem Khronos Group složeným z mnoha předních společností z oblasti IT majících zájem na společném a platformě nezávislém rozhraní, které by umožnilo interaktivní a hardwarově akcelerované vykreslování 3D grafiky ve webových prohlížečích bez potřeby plug-inů. Rozhraní WebGL je dle specifikace dostupné přes standardní DOM rozhraní webových prohlížečů. DOM (Document Object Model) je specifikace platformě a jazykově nezávislého rozhraní pro manipulaci s objekty v HTML a XML dokumentech implementované a programátorům dostupné např. v jazyce JavaScript.

DOM mj. umožňuje na těchto objektech volat metody a dynamicky je modifikovat po načtení webové stránky např. v návaznosti na události vyvolané uživatelem. WebGL je navrženo pro programovací jazyky s automatickou správou paměti jako např. JavaScript, což je jeho hlavní odlišnost od API OpenGL ES 2.0, ze kterého jinak vychází. Na OpenGL ES 2.0 proto také prohlížeče delegují programová volání WebGL aplikací, pokud je toto API k dispozici na grafické kartě. Případně mohou být tato volání přeložena na volání konkurenčního Direct3D API od společnost Microsoft jako si klade za cíl např. projekt ANGLE použitý jako výchozí implementace backendu WebGL v prohlížečích Google Chrome a Mozilla Firefox na operačních systémech Microsoft Windows.

# 2 Analýza a návrh




(a) Exploze modelu pro odkrytí fokusu (vyznačeného červeně) (b) Plně explodovaný model

**Obrázek 2** – *Ukázka dvou variant ilustrativního explodování.* Zdroj: [2].

## 2.1 Problematika ilustrativního explodování

### 2.1.1 Požadavky

Záměrem ilustrativního explodování je odkrýt vybraný díl modelu, nazvaný fokus, odsunutím ostatních dílů, které jej obklopují způsobem, jako kdybychom model postupně rozebírali. Nutnost odsunutí jednoho dílu, aby mohl být odsunut díl druhý, nazýváme *blokující závislost* na tomto druhém dílu. Model může být explodován jen do té míry, aby byl viditelný fokus, anebo kompletně až do posledního dílu. Tento rozdíl ilustruje ***Obrázek 2***. Je žádoucí, aby technika explodování splňovala níže uvedené požadavky.

- Zvolený díl by měl být po odkrytí plně viditelný z pohledu kamery. K lepšímu rozpoznání dílu pomůže, pokud kolem něj bude po odkrytí určitý prostor.
- Při průběhu explodování a po jejím dokončení by mělo být stále snadno domyslitelné původní umístění dílů.

- Odsunutí okolních dílů by mělo být provedeno způsobem, který je fyzicky uskutečnitelný. Díly by tedy neměly v průběhu animace procházet skrz sebe.
- Díly by neměly být explodovány do příliš velké vzdálenosti, aby unikly z pohledu kamery, anebo vyžadovaly oddálení pohledu a tím zmenšení fokusu.
- Průběh animace je možné zkrátit nalezením dílů, které je možné odsunout současně. Je však třeba dbát na dodržení předchozích požadavků.

### 2.1.2 Graf exploze

Graf exploze je datová struktura, z níž je možné pro díl, který chceme odkrýt, vyčíst směry, pořadí a případně vzdálenosti odsunu dílů jej obklopujících. Automatický výpočet grafu exploze na základě vstupního modelu byl již implementován původní aplikací [1]. Na základě analýzy popsané v následující sekci jsme ale přistoupili k menším úpravám původního algoritmu výpočtu.

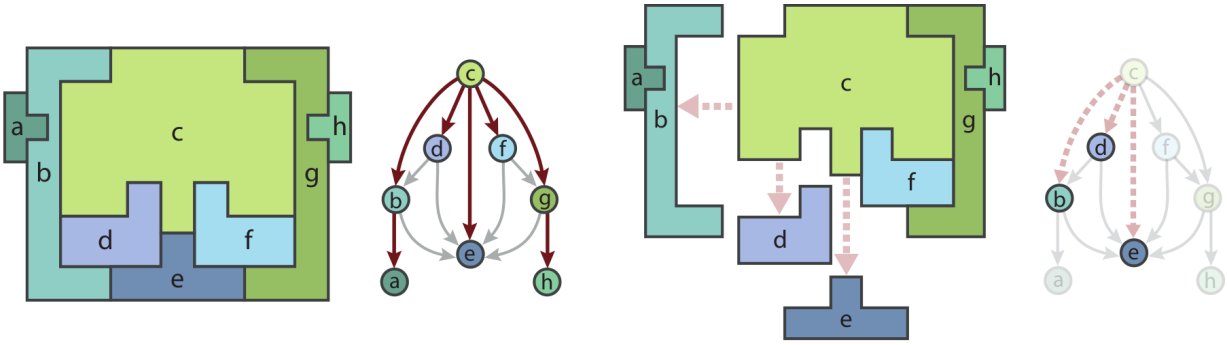
Popíšeme si sémantiku struktury grafu exploze, která v naší práci zůstala nezměněna. Jedná se o acyklický graf, jehož uzly představují jednotlivé díly a hrana směřující vždy od předka k potomkovi vyjadřuje nutnost odsunutí potomka pro uvolnění předka. Pokud tedy budeme chtít odkrýt fokus, tak nejprve sestoupíme po jeho potomcích až k listům, které odebereme jako první v jejich předem vypočteném směru odsunu. Odebráním všech listů určitého potomka, se tento potomek stane sám listem a bude odebrán jako následující. Pořadí odsouvání dílu tedy odpovídá průchodu podgrafu do hloubky (depth-first) s počátkem v uzlu, který představuje fokus. Po odsunutí všech dílů podgrafu zbývá ještě odsunout ostatní díly grafu. Jelikož jsme odebrali všechny potomky fokusu, tak může být uvolněn samotný fokus, avšak ten zůstane na místě a místo něj budou odsunuty všechny zbývající díly ve směru opačném ke směru odsunu fokusu. ***Obrázek 3*** na názorném modelu ukazuje, jakým způsobem je graf exploze interpretován, pokud chceme odkrýt vybraný díl.

Datová struktura kromě samotného grafu s uvedenými údaji uchovává také konfiguraci, se kterou byl graf vygenerován. Tato konfigurace sestává z následujících informací.

- *Množina možných směrů odsunutí* – Tato množina určuje, pro jaké směry bude testována možnost neblokovaného odsunutí dílů.
- *Parametry kompenzující nepřesnosti v geometrii modelu* – Jejich bližší popis není předmětem našeho zájmu a zabývá se jimi původní práce.

Uložením grafu exploze do souboru je možné předejít nutnosti jeho opakovaného výpočtu při každém načtení modelu zvoleného k prohlížení. Výpočet grafu exploze totiž může být zvláště pro složité modely časově náročný, přestože je v našem případě značně urychlen metodou využívající hardwaru grafické karty

navrženou a implementovanou v původní práci.




**Obrázek 3** – *Blokující závislosti mezi díly vyjádřené grafem exploze a jeho interpretace. Pokud chceme odkrýt díl* **d** *musíme nejprve odsunout jeho potomky* **a**, **b** *a* **e**. Zdroj: [2].

## 2.2 Rozbor původní práce

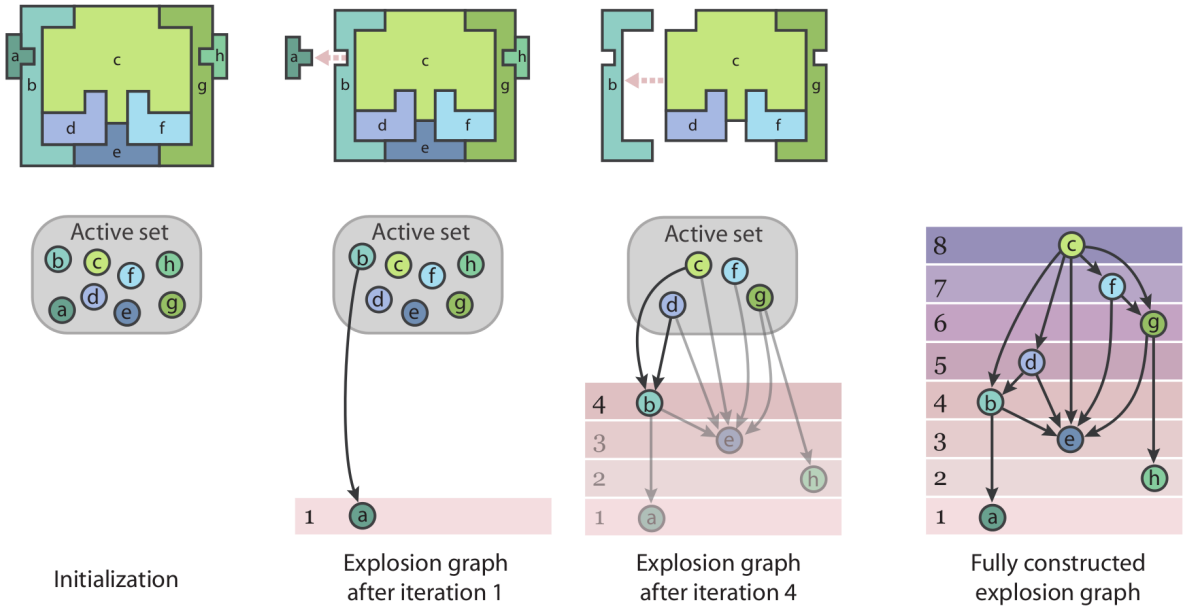
Aplikaci, která je výstupem původní práce [1], lze rozdělit na část zajišťující výpočet grafu exploze a část využívající graf exploze pro výpočet výsledné animace v rámci interaktivního prohlížeče s načteným modelem.

### 2.2.1 Výpočet grafu exploze

Původní aplikace [1] využívá pro výpočet grafu exploze dále popsaný algoritmus vycházející z dřívější práce [3]. V sekci 2.2.3 nicméně dospějeme k tomu, že takto sestavený graf nezachycuje veškeré blokující závislosti mezi díly. Než přistoupíme k samotnému algoritmu, definujeme si pojem *obálka objektu*. Jedná se o jednoduché těleso obepínající daný objekt, které se používá především pro urychlení výpočtu detekce kolize mezi dvěma objekty. V našem případě je tímto tělesem osově zarovnaný kvádr.

Graf exploze je sestavován směrem zdola nahoru postupným přidáváním jednotlivých dílů a orientovaných hran mezi nimi. Na začátku definujeme množinu aktivních dílů $S$, která je naplněna všemi díly modelu. Z množiny $S$ je sestavena podmnožina $P \subseteq S$ obsahující veškeré díly, které nejsou blokovány ostatními díly z množiny $S$ alespoň v jednom z definovaných směrů. Pro každý z těchto dílů a pro každý nalezený směr jeho volného odsunu je vypočtena vzdálenost, o kterou je třeba díl v daném směru odsunout, aby jeho obálka unikla z obálky dílů, jichž se dotýká. Díl s nejmenší takovou vzdáleností je odebrán z množiny $S$ a přidán do grafu exploze jako potomek všech dílů z množiny $S$, kterých se tento díl dotýká. Spolu s ním je do grafu uložena vypočtená vzdálenost a směr odsunu, které nazýváme *vzdálenost* a *směr úniku*. Uvedený postup výběru podmnožiny $P$ a detekce dílu, který je možné z

množiny $S$ odebrat, je opakován, dokud není množina S prázdná, anebo již není možné nalézt směr volného odsunu pro žádný ze zbývajících dílů. Vybrané iterace postupného sestavování grafu exploze pro ukázkový model jsou znázorněny na ***Obrázek 4***.

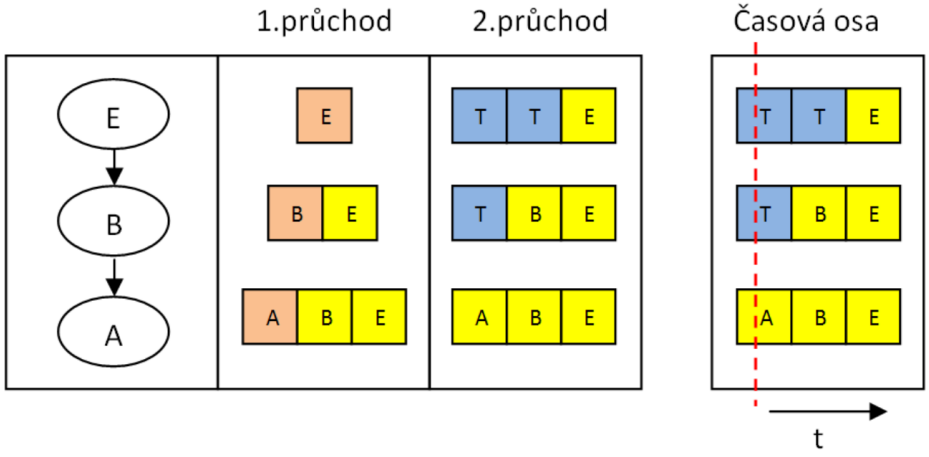



**Obrázek 4** – *Příklad stavů algoritmu pro výpočet grafu exploze ve vybraných iteracích.* Zdroj: [3].

### 2.2.2 Výpočet animace exploze

V předchozí sekci 2.2.1 jsme si popsali, jaké informace nám poskytuje graf exploze. Nyní se zaměříme na to, jak je v původní aplikaci určeno přesné časování animace odsouvání jednotlivých dílů modelu. V sekci 2.1.2 jsme zmínili, že pro odkrytí fokusu je třeba začít odsouváním jeho nejvzdálenějších potomků a dále postupovat grafem exploze směrem k fokusu. Aplikace pracuje tak, že odsouvání potomka fokusu je zahájeno v momentě dokončení animace odsunutí jeho přímých potomků o jejich vzdálenost úniku definovanou v sekci 2.2.1. Přímí potomci zase čekají na dokončení animace úniku jejich přímých potomků. Podnět na animaci přímých potomků je jim vyslán jejich předkem ve stejný moment, takže jsou všichni animováni současně, jakmile jejich potomci dokončí svou animaci. Odsunutím dílu o jeho vzdálenost úniku však jeho animace nekončí. Je třeba dále zajistit, aby takto odsunutý díl nepřekážel v odsouvání následujících dílů, které postupně odkrývají fokus. Tento problém je v původní aplikaci řešen následujícím způsobem, avšak v sekci 2.2.3 si ukážeme, že toto řešení problému vždy nezabrání.

Algoritmus navržený v původní práci [1] pracuje tak, že spolu s momentálně odsouvaným dílem jsou o stejnou vzdálenost současně odsouváni i jeho potomci v grafu exploze, pokud mají stejný směr úniku. V případě, že stejný směr úniku nemají, tak pouze čekají. Pro docílení současného odsouvání potomků s jejich předky původní práce zavádí koncept tzv. *animačních bloků*. Každý blok reprezentuje konstantní časový interval s animačními informacemi skládajícími se ze směru a vzdálenosti odsunu. Každý díl si nese seznam těchto animačních bloků, kde daný blok díl buďto zdědil od svých předků, je jeho vlastní, anebo jde o speciální vyčkávající animační bloky, které představují pouze časové intervaly, kdy se díl nehýbe. Seznam animačních bloků daného dílu určuje posloupnost dílčích kroků animace, které je třeba ve správný moment vykonat pro uskutečnění jeho exploze. Vlastní animační bloky určitého dílu obsahují směr a vzdálenost, o kterou má být tento díl odsunut v momentě jeho uvolnění ze zbývající části modelu. Jedná se o směr a vzdálenost úniku uloženou v grafu exploze spolu s daným dílem. Zděděné animační bloky obsahují směry a vzdálenosti odsunu, které díl zdědil od svých předků a zajišťují současný pohyb s předky v momentě jejich uvolnění. Animační bloky jsou na potomky kopírovány v prvním průchodu grafu do hloubky. Pokud ovšem zděděný animační blok obsahuje jiný směr odsunu než vlastní animační blok, je zděděný animační blok nahrazen vyčkávajícím animačním blokem, což představuje situaci, kdy potomek nemusí během exploze uvolnit prostor před svým předkem, protože ten je odsouván v jiném směru. V druhém průchodu grafu jsou na začátky seznamů pouze doplněny vyčkávající animační bloky, čímž je dosaženo jejich časového zarovnání. Uvedený postup ilustruje ***Obrázek 5***.




**Obrázek 5** – *Příklad vytváření seznamu animačních bloků. Uvažujeme díl* E*, který chceme odkrýt a díly* B *a* A*, které jsou jeho potomky a blokují jej. Každý čtvereček představuje jeden animační blok. Průchodem grafu od dílu* E *dolů jsou animační bloky kopírovány na jeho potomky. Vlastní animační blok je označen oranžově, zděděné bloky žlutě a vyčkávající bloky modře.* Zdroj: [1].

Dosud jsme se zabývali pouze animací dílů, které jsou potomky fokusu, ale pro jeho úplné odkrytí je třeba odsunout i zbývající díly, tedy jeho sourozence a předky. V původním algoritmu [1] jsou všechny tyto díly odsouvány najednou stejným směrem a to opačným ke směru úniku fokusu, který zůstává nehybný. V následující sekci 2.2.3 si ale ukážeme, že takovýto přístup může způsobovat problémy.

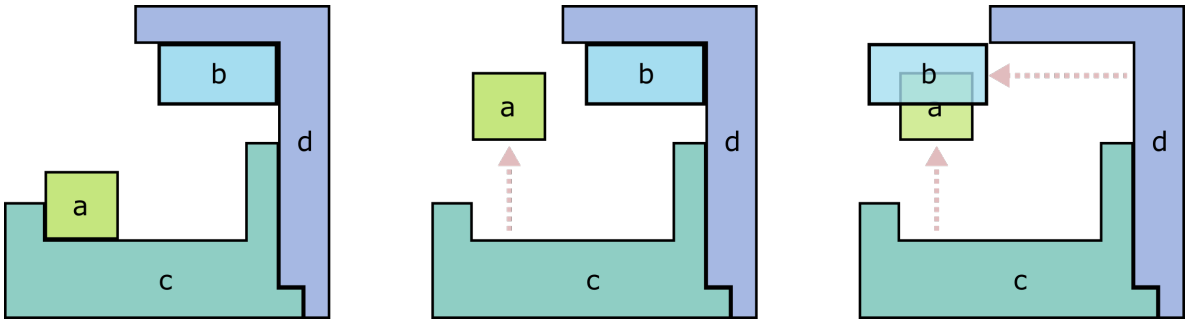
### 2.2.3 Problémy v původní aplikaci

V této práci rozebíráme problémy původní aplikace [1] pouze z hlediska požadavků definovaných v sekci 2.1.1. Jak je naznačeno v závěru původní práce, během animace exploze se mohou vyskytnout dva nežádoucí jevy.

### 2.2.4 Procházení dílů skrz sebe

Prvním nežádoucím jevem je, že může docházet k procházení dílů skrz sebe. Autor práce navrhuje zaměřit se na tzv. globální blokující omezení, tedy situace, kdy nějaký díl blokuje volný odsun jiného dílu, kterého se však nedotýká. Dále je v závěru zmíněno, že tato globální omezení nemohou být nalezena v grafu exploze, jehož výpočtem jsme se zabývali v sekci 2.2.1. Problém je v tom, že přestože jsme díly do grafu exploze přidávali v pořadí možného postupného fyzického rozebrání daného modelu, nezanesli jsme toto pořadí do grafu exploze úplně. Pokud bychom měli toto pořadí respektovat úplně, potřebovali bychom místo grafu uspořádaný seznam exploze. Z grafu můžeme pouze vyčíst, že před odsunutím určitého dílu je třeba nejprve postupně odsunout jeho předcházející sourozence a poté jeho přímé potomky od prvního k poslednímu, ale již nevíme v jakém pořadí odebírat potomky druhé a vyšší úrovně napříč různými větvemi. Jinými slovy, pořadí střídavého přidávání potomků k různým větvím je v grafu ztraceno. Příčina pozorovaných kolizí mezi díly nicméně nespočívá jen v nedokonalém algoritmu pro výpočet grafu exploze, ale také ve výpočtu animace exploze. V následující sekci 2.3 navrhneme alespoň částečné řešení této příčiny kolizí, která umožňuje využívat stávající grafy exploze.

Výpočet animace exploze popsaný v sekci 2.2.2 nezohledňuje blokující závislost, kterou je možné vyčíst z pořadí sourozenců. Jak jsme uvedli, všichni sourozenci jsou totiž odsouváni ve stejný moment nehledě na jejich směr úniku, což mezi nimi může způsobit kolize, pokud se jejich směr úniku vzájemně liší. Také jsme zmínili, že veškeré díly zbývající po odsunutí všech potomků fokusu, tedy i jeho sourozenci, jsou odsouvány také najednou a to ve směru opačném ke směru úniku fokusu. Zde tedy opět není zohledněno pořadí sourozenců a může docházet ke kolizím jak mezi nimi navzájem, tak mezi nimi a fokusem.



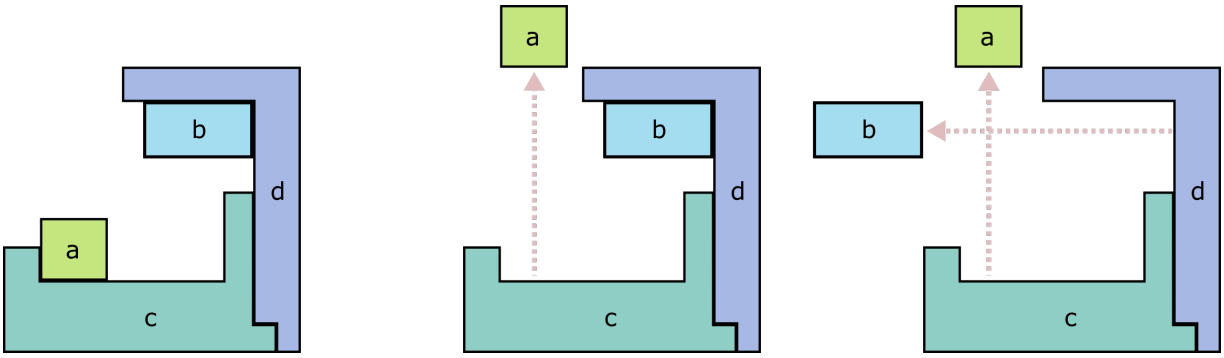

**Obrázek 6** – *Ukázka kolize způsobené nedostatečnou vzdáleností úniku.*

Analýzou algoritmu pro výpočet animace exploze jsme narazili na další situaci, kdy může dojít ke kolizi a to mezi právě odsouvanými díly a díly, které byly odsunuté před nimi. Algoritmus pro výpočet grafu exploze se těmito kolizemi nemusí zabývat, pokud předpokládá, že díly budou odsunuty vždy dostatečně daleko, aby zanechaly prostor pro následující odsouvané díly. Graf exploze však může poskytnout pomocné informace pro stanovení vhodné výsledné vzdálenosti odsunu. V původní práci je touto pomocnou informací přidruženou ke každému uvolněnému dílu již zmíněná vzdálenost úniku, která je definována jako vzdálenost úniku obálky tohoto dílu ve směru jeho nejkratšího takového úniku, z obálky dílů, kterých se dotýkal než byl uvolněn.

V sekci 2.2.2 zmiňujeme, že pokud má potomek stejný směr úniku jako jeho předek, tak je na něj přenesen animační blok tohoto předka a nemůže se tedy s ním dostat do kolize, neboť jsou pohybováni současně. Může se však dostat do kolize s předky, kteří mají jiný směr úniku nebo s díly, kteří nejsou jeho předci, pokud v momentě jejich odsouvání bude ponechán pouze své vzdálenosti úniku tak, jak je definována v původní práci, kdy bude považován za již odebraný. Tato vzdálenost nemusí být dostatečná, aby kolizím zamezila, jak ilustruje ***Obrázek 6***.

### 2.2.5 Zastínění fokusu

Druhý nežádoucí jev opět souvisí se zvoleným výpočtem animace exploze. Výsledná vzdálenost odsunu dílu dosažená provedením všech animačních bloků totiž nijak nezohledňuje pohled kamery. Z určitého pohledu se tak fokus může jevit jako zastíněný okolními odsunutými díly.




**Obrázek 7** – *Zamezení kolize změnou definice vzdálenosti úniku.*

## 2.3 Návrh vylepšení původní práce

### 2.3.1 Změny ve výpočtu animace exploze

Pro zamezení některých příčin nalezených problémů dostačuje pouze úprava výpočtu animace exploze a není třeba měnit algoritmus pro výpočet grafu exploze. V sekci 2.2.4 poukazujeme na to, že není úplně využit potenciál dostupného grafu exploze, co se týče zabránění některých předpokladů stojících za nežádoucím jevem procházení současně odsouvaných dílů skrz sebe.

Navržené řešení spočívá v tom, že sourozenci v grafu exploze nejsou z modelu odebíráni současně, ale s vzájemným opožděním o jeden vyčkávající animační blok, podobně jako předci vůči svým potomkům. Každý díl je tedy z modelu uvolněn až poté, co je uvolněn jeho předešlý sourozenec, což více odpovídá sekvenci, ve které byl graf exploze sestavován.

### 2.3.2 Změny ve výpočtu grafu exploze

Graf exploze nám poskytuje ke každému uvolnitelnému dílu informaci o jeho vzdálenosti úniku, ale situace na ***Obrázek 6*** ze sekce 2.2.4 názorně ukazuje, že tato vzdálenost není dostačující pro zamezení kolizí s následujícími odsouvanými díly. Navržená úprava grafu exploze tedy spočívá v nahrazení této vzdálenosti o vzdálenost úniku obálky odebíraného dílu z obálky všech ostatních dosud neodebraných dílů, nikoliv jen dílů, kterých se dotýká. Uplatnění nové definice vzdálenosti úniku zachycuje na stejné situaci ***Obrázek 7***.

Zamezení ostatních příčin procházení součástek skrz sebe by vyžadovalo rozsáhlejší úpravy v algoritmu pro výpočet grafu exploze, které jsme ve výsledné aplikaci ale neimplementovali. Nastíníme si je alespoň teoreticky v následujícím odstavci, kde se odkazujeme na algoritmus popsaný v sekci 2.2.1.

Jedním z řešením by bylo při generování grafu exploze sledovat, jaké díly přibyly do množiny neblokovaných dílů $P$ po odebrání vybraného dílu z této množiny a jeho přidání do grafu. Díly přibylé do množiny $P$ by byly takové, které posledně odebraný díl zastiňoval, a tak přidáním odebraného dílu do grafu exploze jako potomka těchto přibylých dílů bychom dosáhli toho, že by musely být odsunuti před ním. Tato informace se v současném algoritmu ztrácí, neboť zaznamenáváme pouze blokující závislosti mezi dotýkajícími se díly.

### 2.3.3 Současná animace více dílů

Jedním z požadavkem ilustrovaného explodování je, aby exploze byla co nejjednodušší a uživatel nemusel být rozptylován postupnou explozí dílů, které není třeba pro uvolnění fokusu odsouvat. Původní koncept animačních bloků zajišťuje, že předci jsou odsouvání o jeden vyčkávající animační blok později než jejich potomci, jak popisuje sekce 2.2.2 a názorně ilustruje ***Obrázek 5***. Tato skutečnost způsobuje postupné explodování modelu od potomků k předkům, které nemusí být vždy žádoucí.

V naší práci navrhujeme neopožďovat animaci předka vůči jemu posledně odsunutému přímému potomkovi, pokud je jeho směr úniku rovnoběžný se směrem úniku tohoto potomka. Animace předka by totiž dle původního algoritmu následovala ihned po skončení animace tohoto potomka a pokud je jejich směr odsunu stejný, tak je možné tyto dva díly považovat za jeden celek. Pokud je jejich směr odsunu opačný, tak se díly rozbíhají na opačné strany a nemůže tedy mezi nimi nastat kolize. V obou případech rovnoběžného směru úniku je tedy možné tyto dva díly animovat současně. V původní práci jsou současně odsouváni všichni sourozenci, ale v sekci 2.2.4 jsme poukázali na to, že to jedna z příčin procházení dílů skrze sebe, a tak jsme v sekci 2.3.1 navrhli opozdit animaci sourozenců o jeden vyčkávající animační blok. Tím se exploze stává více sekvenční, což alespoň částečně vykompenzujeme tím, že současnou animaci založenou na rovnoběžném směru úniku dvou dílu uplatníme i mezi sourozenci. Díl, který má rovnoběžný směr úniku se směrem úniku jeho předchozího sourozence, který má být dle grafu exploze odebrán před ním, je tedy animován současně s ním.

### 2.3.4 Zamezení vznášejících se dílů

Během explodování může nastat situace, kdy odsunutím nějakého dílu, zbyde vedle odkrytého fokusu ještě další díl, který se fokusu ani ostatních dílů nedotýká a působí, jakoby se vznášel ve vzduchu. Tento problém řešíme upravením algoritmu pro výpočet grafu exploze tak, aby předně odebíral díly, které po svém odebrání po sobě zanechají co nejmenší počet vznášejících se skupin součástek.

## 2.4 Serverová část

### 2.4.1 Webová platforma

Předpokladem webové aplikace je webový server vytvářející prostředí, ve kterém může být serverová část aplikace provozována. K dispozici je mnoho webových serverů podporujících různé programovací jazyky a jejich frameworky, avšak nejvhodněji se jevila volba webového serveru založeného na platformě Java, který by umožňoval vytvoření aplikace, jenž by bylo možné jednoduše provázat s již hotovou implementací desktopové Java aplikace.

Pro tento účel byl zvolen open-source webový server Apache Tomcat verze 8.0.15 vyvíjený jako svobodný software otevřenou komunitou pod dohledem Apache Software Foundation. Vzhledem k základním požadavkům aplikace by ale byl stejně tak dostačující i jiný méně robustnější server.

Aplikace je implementovaná ve Spring Frameworku verze 4.0.1 využívajícím webové API a běhové prostředí platformy Java EE 7. Verze 7 byla zvolena kvůli její podpoře specifikace pro tvorbu uživatelského rozhraní Java Server Faces (JSF) verze 2.2, která oficiálně podporuje a usnadňuje používání značkovacího jazyka HTML5. V rámci JSF je použit moderní šablonovací systém Facelets, který zjednodušuje kompozici a znovupoužitelnost značkovacího kódu v podobě XML dokumentů a jeho provázání s webovými komponentami. Pro doplnění technologie JSF byla použita knihovna OmniFaces verze 1.10, která poskytuje řešení obvyklých nedostatků JSF a knihovna PrimeFaces verze 5.1 rozšiřující JSF převážně o rozsáhlou sadu komponent s užitečnou funkcionalitou (např. AJAX). Obě jsou však v aplikaci použity velmi zřídka. Knihovny jsou dostupné jako svobodný software.

### 2.4.2 Grafické výpočty

Výpočty související s 3D grafikou jsou realizovány prostřednictvím knihovny JOGL verze 2.2.4, která na nízké úrovni zpřístupňuje OpenGL API a navíc nabízí pomocné vzory a nástroje pro implementaci grafického rozhraní a jeho ovládání pomocí vstupních zařízení.

## 2.5 Klientská část

Použití značkovacího jazyka HTML5 pro klientskou část aplikace vychází z jeho standardizace značkovacích elementů a rozhraní technologie WebGL, které je přístupné přes skriptovací jazyk JavaScript. Jelikož WebGL v podstatě mapuje nízkoúrovňové rozhraní OpenGL ES 2.0, byla pro implementaci prohlížeče modelů zvolena JavaScriptové knihovna, která přidáním vrstvy abstrakce zjednodušuje některé rutiny spojené s vykreslováním 3D grafiky a interakcí s uživatelem pomocí

myši a klávesnice. Zároveň však bylo potřeba zachovat dostatečný výkon i pro zpracování objemných modelů a poskytnout přístup k objektům tvořících model, které je potřeba animovat.

Zvolená knihovna se jmenuje Three.js a je vyvíjena jako svobodný software pod licencí MIT. Kromě uvedených vlastností byla vybraná z důvodu poměrně rozsáhlé funkcionality a dostupnosti podpůrných nástrojů pro konverzi modelů z textového formátu OBJ používaného desktopovou aplikací do formátu JSON s přidruženými binárními vertexovými daty v souboru BIN. Tato kombinace je datově menší a zároveň rychleji zpracovatelná klientem vzhledem k tomu, že s uvedeným binárním formátem přímo pracuje grafická karta bez nutnosti konverze a JSON je možné nativně parsovat v JavaScriptu. Nicméně stejně tak by bylo možné použít např. i knihovnu GLGE podporující komprimovaný binární formát OpenCTM.

Především pro usnadnění animace a jiné manipulace s DOM elementy byla vybrána populární knihovna jQuery verze 1.11.2, která poskytuje jednotné API kompatibilitní se stále nezanedbatelně rozšířenými prohlížeči Internet Explorer verze 6, 7 a 8. Kompatibilitu s HTML5 ve starších prohlížečích dále zajišťuje knihovna nazvaná HTML5 Shiv (iehtml5.js).

## 2.6 Optimalizace aplikace

### 2.6.1 Rychlost zpracování modelu prohlížečem

Jakmile uživatel zvolí model k prohlížení, mělo by jeho zpracování v prohlížeči proběhnout v co možná nejkratším čase. Zpracováním je myšlen proces načtení všech potřebných souborů definujících daný model, interpretace dat v nich obsažených a jejich nahrání v příslušném datovém formátu na grafickou kartu. Doba potřebná pro vykreslení dat nahraných na grafickou kartu zde není uvažována, neboť se odvíjí od efektivity vykreslovacího algoritmu a úrovně efektů výsledného zobrazení, což jsou parametry společné pro všechny testované formáty.

Rychlost zpracování se odvíjí od množství informace definující model, tedy především od složitosti jeho geometrie co do počtu vertexů jakožto největší složky této informace, a dále od formátu dat modelu. Předpokladem bylo zachování veškeré této informace, proto byla vybrána optimalizace cestou volby takového formátu dat modelu, který by po obdržení modelu klientem ze serveru umožnil jeho rychlé zpracování ve webovém prohlížeči i v případě geometricky složitých modelů.

### 2.6.2 Popis formátů dat modelu

Formát dat modelu lze pro účely testu rozdělit do dvou úrovní. Za prvé formát na úrovni souboru obsahujícího definici složek modelu v podobě organizace jeho

geometrických a materiálových dat, a za druhé formát na úrovni samotných geometrických a materiálových dat. Tato data mohou být buďto přímo v souboru s definicí modelu, anebo mohou být uložené v samostatných souborech odkazovaných z tohoto souboru.

#### 2.6.2.1. Testované varianty formátů

1. *JSON* – JSON je textový souborový formát obsahující definici složek modelu, geometrická i materiálová data. Tato varianta byla vybrána z důvodu nativní implementace parseru souborů JSON v JavaScriptových enginech webových prohlížečů.

2. *JSON + BIN* – JSON je textový souborový formát obsahující definici složek modelu a materiálová data, což dohromady představuje nejmenší část informace definující celý model. BIN je binární souborový formát obsahující geometrická data ve stejném datovém formátu, jaký používají geometrické buffery ve WebGL, takže je možné je do nich přímo vkopírovat bez nutnosti datové konverze. Tato varianta byla vybraná jako reprezentant této přednosti binárního souboru.

3. *OBJ + MTL* – Wavefront OBJ je textový souborový formát obsahující definici složek modelu, geometrická data a případně odkaz na jeden nebo více přidružených MTL souborů. MTL je textový souborový formát obsahující materiálová data. Tato varianta byla vybrána, protože se jedná o jeden z nejrozšířenějších výměnných formátů a kvůli jeho jednoduchosti.

### 2.6.3 Faktory ovlivňující rychlost zpracování modelu u klienta

Tyto dva faktory vycházejí z dvou výše uvedených úrovní formátů.

1. Efektivita algoritmu pro parsování souboru obsahujícího data modelu a sestavení jeho reprezentace v paměti v podobě JavaScriptového objektu. Tento faktor závisí na formátu souboru definujícím složky modelu (JSON nebo OBJ).

2. Efektivita algoritmu pro naplnění vertex array bufferů geometrickými daty z objektové reprezentace modelu v paměti. Tento faktor závisí na volbě formátu geometrických dat (JSON, OBJ + MTL nebo BIN).

### 2.6.4 Výsledky testovaní vybraných variant formátů dat modelu

**Tabulka 1** – *Počet vertexů exemplárních modelů použitých pro testování.*

| Název modelu | Počet vertexů |
|---|---|
| cow_triangles | 3 100 |
| head_anatomy4 | 97 141 |
| gearbox | 334 540 |

**Tabulka 2** – *Datová velikost exemplárních modelů ve vybraných variantách formátů.*

| Název modelu | Datová velikost [B] | | |
|---|---|---|---|
| | 1. JSON | 2. JSON + BIN | 3. OBJ + MTL |
| cow_triangles | 359 452 | 1 603 + 197 464 | 394 892 + 702 |
| head_anatomy4 | 13 120 397 | 3 901 + 6 458 344 | 13 835 368 + 1 723 |
| gearbox | 35 900 045 | 2 814 + 17 432 756 | 37 852 307 + 1 231 |

**Tabulka 3** – *Rychlost zpracování exemplárních modelů prohlížečem ve vybraných variantách formátů.*

| Název modelu | Čas [ms] | | |
|---|---|---|---|
| | 1. JSON | 2. JSON + BIN | 3. OBJ + MTL |
| cow_triangles | 90 | 269 | 162 |
| head_anatomy4 | 2547 | 1478 | 4547 |
| gearbox | 205339 | 6924 | 14268 |

### 2.6.5 Závěr testování

Z testování vyplývá, že nejoptimálnější je 2. varianta (JSON + BIN), která již v případě modelu „head_anatomy4" s poměrně malým počtem vertexů (97 141)

dosahuje v porovnání s ostatními variantami nejvyšší rychlosti.

# 3 Popis realizovaného řešení

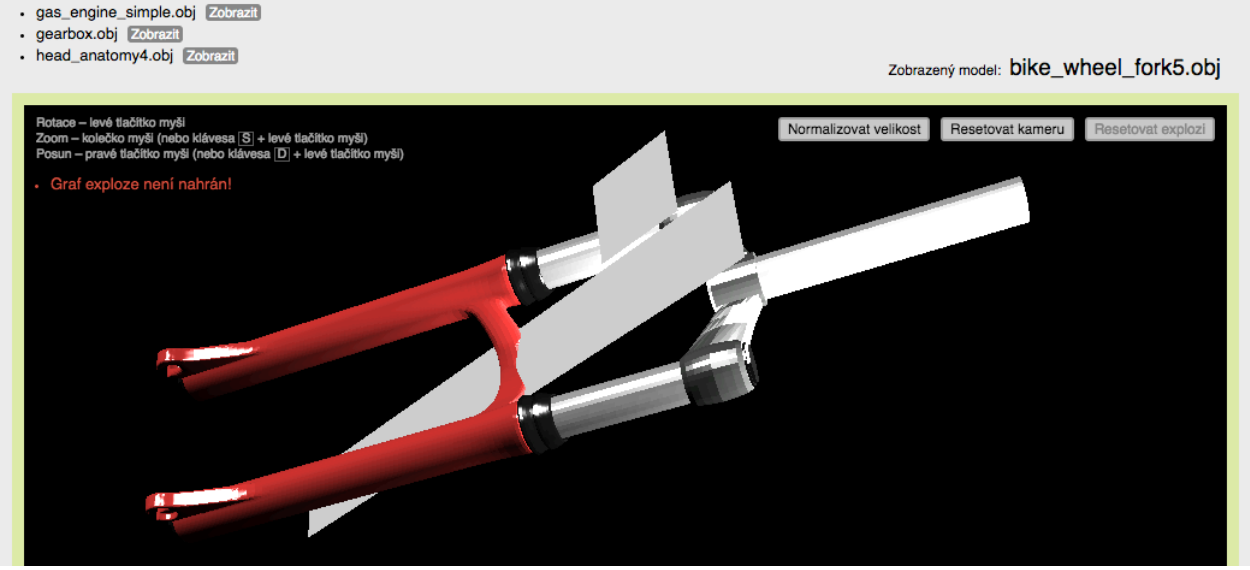
## 3.1 Uživatelské rozhraní

Webová aplikace je rozdělena do následujících tří sekcí přístupných přes položky navigačního menu.

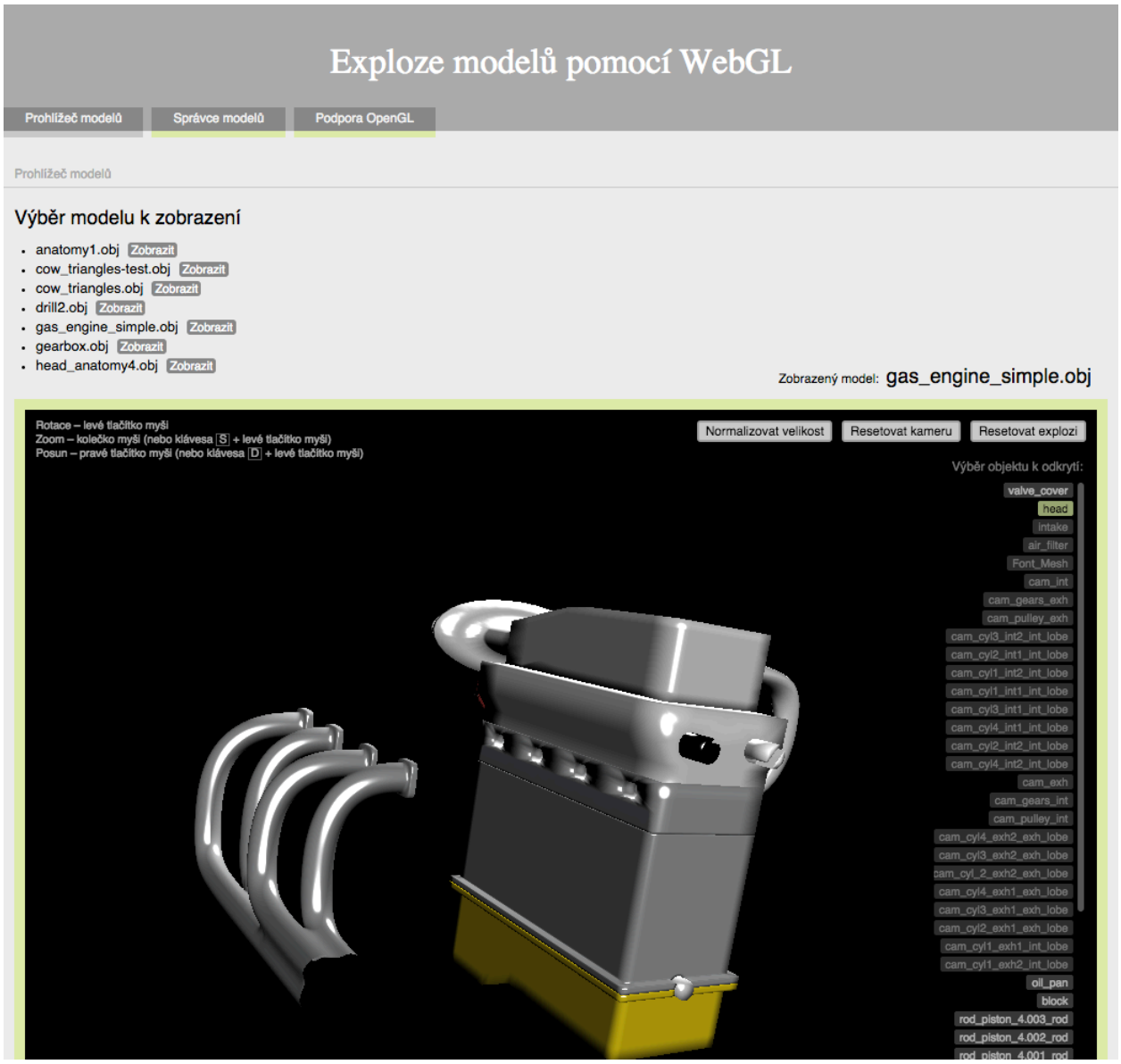
### 3.1.1 Prohlížeč modelů

Tato stránka obsahuje výpis nahraných nebo převedených souborů s modely, které je možné přímo zobrazit v projekčním okně prohlížeče modelů nacházejícím se pod výpisem.

Nad oknem prohlížeče modelů je uveden název momentálně zobrazeného modelu. V horní části prohlížeče modelů překrývají scénu tlačítka pro: (a) normalizaci velikosti modelu (pro případ, že je nahrán příliš rozměrný model), (b) resetování kamery do původního čelního pohledu a (c) resetování přehrané animace exploze. Dále je v levém horním rohu nápověda k manipulaci s modelem, pod níž se v případě vadného nebo chybějícího souboru s vypočítaným grafem exploze zobrazují červené chybové hlášky. Normalizace velikosti modelu jej zobrazí v takové velikosti, aby se celým objemem vešel do pohledu z původní pozice kamery, jelikož zobrazené modely můžou mít různá měřítka.




**Obrázek 8** – *Výřez prohlížeče modelů s chybovou hláškou.*




**Obrázek 9** – *Celá stránka prohlížeče modelů s dlouhým seznamem objektů modelu.*

Jakmile je graf exploze úspěšně načtený, zobrazí se v právě části prohlížeče modelů seznam tlačítek s názvy objektů tvořících model, přes která je možné spustit animaci exploze. V případech, kdy je seznam tlačítek příliš dlouhý, se zobrazí skrolovací lišta. Objekty v seznamu, které nemají definovanou animaci exploze se zobrazují méně výrazně a nemění stav při najetí myši.

Během animace exploze není nutné čekat na dokončení animace, ale je možné vybrat jiný objekt k explozi, anebo stisknout tlačítko „Resetovat explozi“, čímž se animace začne ze současné polohy přehrávat pozpátku do počátečního stavu a poté

případně plynule započne exploze jiného zvoleného objektu.

Pro manipulaci s modelem slouží pohyb myši při stisku levého (rotace) nebo pravého (posun, tzv. panning) tlačítka a kolečko myši (přiblížení) popř. stisk klávesy S (přiblížení) nebo D (posun) v kombinaci se stisknutým levým tlačítkem myši.

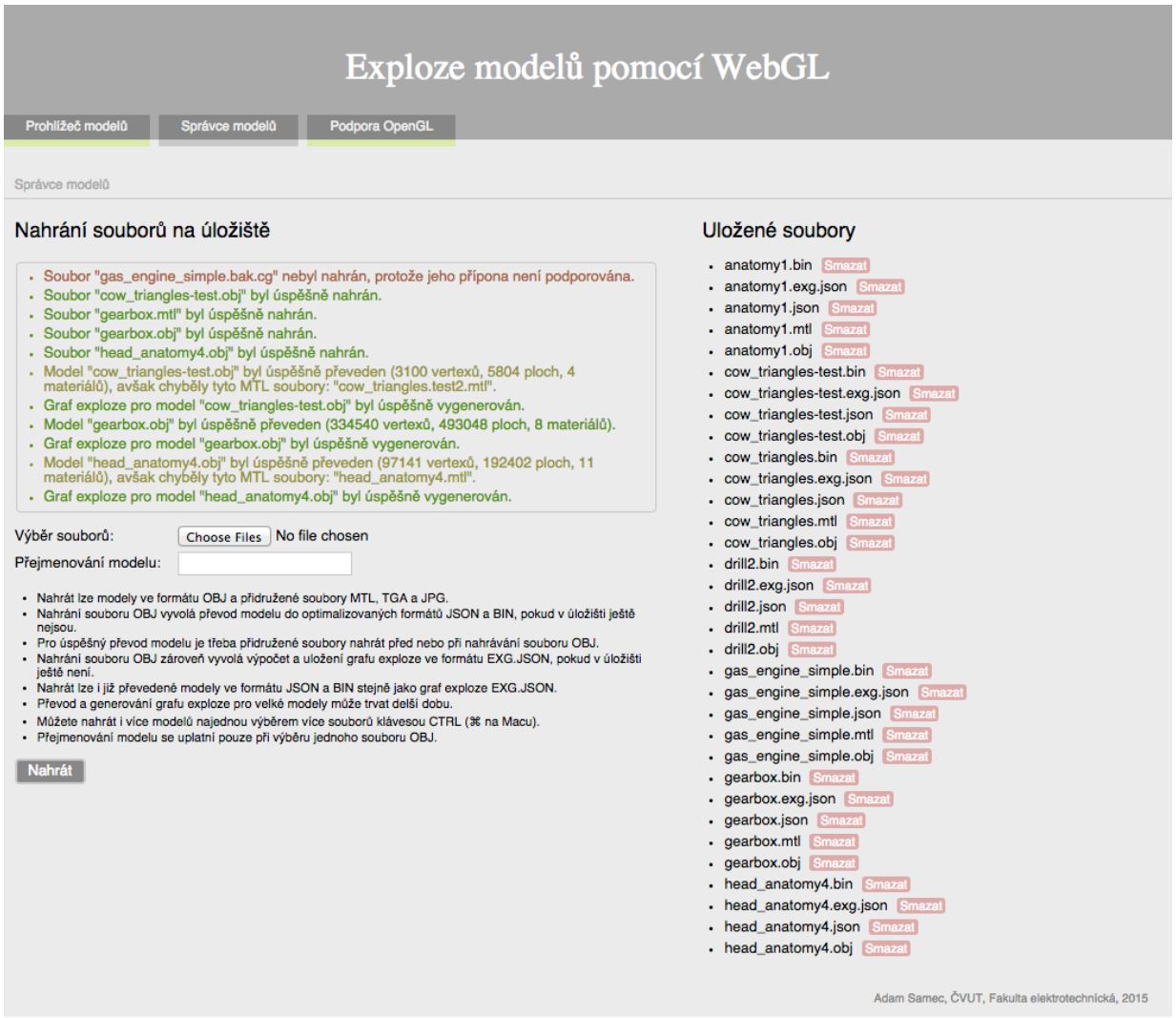
### 3.1.2 Správce modelů

Stránka *Správce modelů* poskytuje jednoduché administrační rozhraní pro nahrávání a mazání souborů uložených na serveru.

#### 3.1.2.1. Nahrávání souborů

Nahrávání souborů je zajištěno validací dat a má tyto vlastnosti:

- Nahrát lze modely ve formátu OBJ a přidružené soubory MTL, TGA a JPG.
- Nahrání souboru OBJ vyvolá převod modelu do optimalizovaných formátů JSON a BIN, pokud v úložišti ještě nejsou.
- Pro úspěšný převod modelu je třeba přidružené soubory nahrát před nebo při nahrávání souboru OBJ.
- Nahrání souboru OBJ zároveň vyvolá výpočet a uložení grafu exploze ve formátu EXG.JSON, pokud v úložišti ještě není.
- Nahrát lze i již převedené modely ve formátu JSON a BIN stejně jako graf exploze EXG.JSON.
- Je možné nahrát i více modelů najednou výběrem více souborů klávesou CTRL (⌘ na platformě Apple Macintosh).
- Přejmenování modelu slouží pro změnu názvu nahraného modelu a uplatní se pouze v případě, že byl k nahraní vybrán jen jediný soubor OBJ.



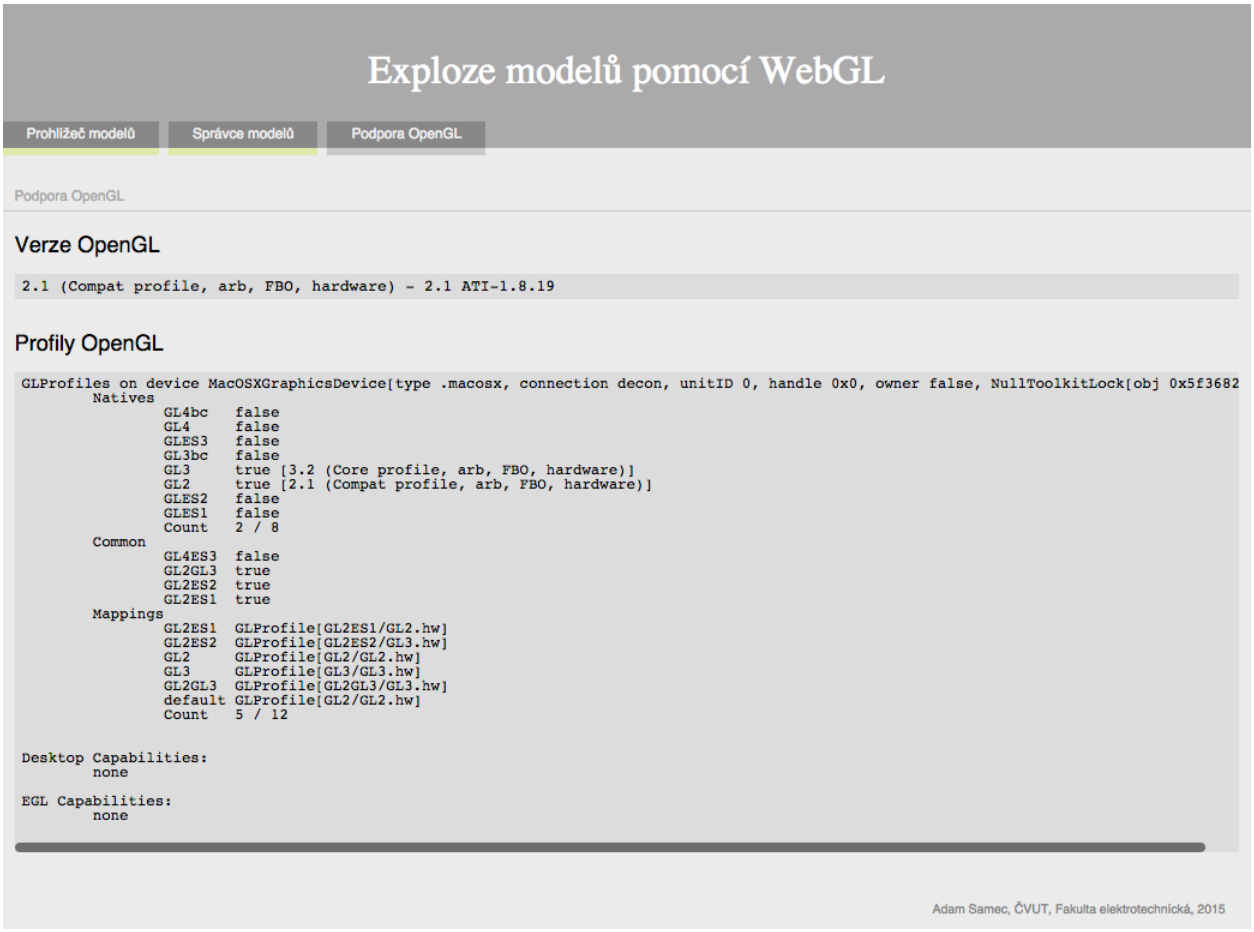

**Obrázek 10** – *Výsledek nahrání souborů přes Správce modelů.*

Výsledek akce nahrání souborů je ohlášen uživateli pomocí zpráv zvýrazněných dle charakteru dané dílčí operace a to zelenou, oranžovou nebo červenou barvou. K této funkcionalitě je použit JSF tag <*h:messages*> využívající třídy *FacesMessage*. Tyto zprávy jsou zároveň s podrobnými informacemi o důležitých operacích a jejími výstupy logovány na standardní výstup webového serveru, což může být v budoucnu snadno nahrazeno logováním do souboru.

#### 3.1.2.2. Mazání souborů

Mazání souborů probíhá bez potřeby načítat celou stránku znovu díky metodě AJAX zajištěné knihovnou PrimeFaces.

### 3.1.3 Podpora OpenGL




**Obrázek 11** – *Výpis informací o podpoře OpenGL dostupné na serveru.*

Stránka Podpora OpenGL slouží k zobrazení informací o podporované verzi a profilech OpenGL ovladačem grafické karty na webovém serveru.

## 3.2 Konverze modelů

Konverze nahraných modelů OBJ do formátu JSON a BIN probíhá vytvořením nového procesu s příkazem spouštějícím skript *convert_obj_three.py* v interpretu jazyka Python. Tento skript je dodávaný s knihovnou Three.js. Převod modelu je spuštěn ihned po nahrání souborů zvolených ve formuláři na server a odpověď webovému prohlížeči je tak pozdržena po dobu dokončení tohoto procesu.

## 3.3 Výpočet a uložení grafu exploze

V případě, že se v dávce nahraných souborů nachází modely OBJ, následuje po jejich konverzi výpočet grafu kolizí a grafu exploze, ale jen když se graf exploze pro daný název modelu na serveru ještě nenachází. Pokud by uživatel chtěl opětovně graf vygenerovat, musí starý graf smazat a nahrát model OBJ na server znovu. Desktopová aplikace je upravena tak, aby ukládala graf exploze do formátu JSON s příponou *.exg.json*, aby jej bylo možné efektivněji načíst v JavaScriptu.

## 3.4 Přizpůsobení desktopové aplikace

Zdrojové soubory již hotové desktopové aplikace organizované v balíčkách byly přímo začleněny do webové aplikace s následujícími nezbytnými úpravami.

Desktopová aplikace byla omezena jen na požadovanou funkcionalitu výpočtu grafu exploze a byla tedy zbavena grafického rozhraní pro interaktivní explodování 3D modelu a tedy i veškerých výpočtů pro vykreslování modelu na obrazovku.

Další zásahy v kódu původní aplikace [1] vyžadovala aktualizace knihovny JOGL z verze 1.1.1 na verzi 2.2.4, kde je jedním ze zásadních rozdílů zavedení OpenGL profilů vyplývající z historického vývoje OpenGL API a dále změny v třídách zajišťujících okenní rozhraní, animaci a zpracování událostí týkajících se vykreslování a polohovacích zařízení. Současně byly nasazeny aktuálnější verze dalších pomocných knihoven pro práci s 3D grafikou dodané vedoucím práce, což si vyžádalo asi největší změny v původním kódu.

# 4 Závěr

Na základě původní diplomové práce a analýzy jiných předchozí prací, jsme si popsali problematiku automatického výpočtu exploze vstupního modelu. Ověřili a odůvodnili jsme si závěr původní práce, že použitý algoritmus pro výpočet grafu exploze nezaručuje dokonalé rozpoznání všech blokujících závislostí mezi díly. Během animace exploze tedy může docházet procházení dílů skrz sebe. Navrhli a implementovali jsme úpravy v původních algoritmech, které alespoň částečně zabraňují tomuto nežádoucímu jevu a navrhli jsme způsob, jak mu zabránit úplně.

Dále jsme zapracovali řešení problému vznášejících se součástek a umožnili v omezených případech současný odsun dílů, které není třeba jednotlivě explodovat, čímž je exploze přehlednější.

Úspěšně jsme upravili původní desktopovou aplikaci do podoby webové aplikace využívající technologii WebGL k prohlížení 3D modelů ve webovém prohlížeči. Výsledná webová aplikace poskytuje požadovanou funkcionalitu, tedy interaktivní prohlížení modelů s možností ilustrativního explodování zvolených dílů a jednoduchou správu modelových souborů.

V budoucnu by interaktivní prohlížeč modelů mohl být rozšířen o ovládací prvky umožňující přizpůsobení rychlosti a vzdálenosti exploze dílů modelu, aby prohlížeč lépe vyhověl požadavkům různých složitostí modelů. Dalším vylepšením by mohl být indikátor průběhu zpracování souborů nahrávaných na server.

Zadání práce jsme splnili až na otestování funkčnosti explodovaného zobrazení na více modelech, čemuž jsme nevěnovali bližší pozornost.

# Seznam použité literatury

[1] UHER, Jiří. *Ilustrativní explodování 3D modelu*. Praha: ČVUT, 2013. Diplomová práce. ČVUT v Praze, Fakulta elektrotechnická, Katedra počítačové grafiky a interakce.

[2] AGRAWALA, Maneesh, Doantam PHAN, Julie HEISER, John HAYMAKER, Jeff KLINGNER, Pat HANRAHAN a Barbara TVERSKY. Designing Effective Step-by-step Assembly Instructions. *ACM Transactions on Graphics*. 2003, roč. 22, č. 3, s. 828–837.

[3] LI, Wilmot W. *Interactive Illustrations for Visualizing Complex 3D Objects*. 2008. Disertační práce. University of Washington, Computer Science & Engineering.

# Návod k instalaci

## Spuštění webové aplikace

Webová aplikace je vyvinuta ve vývojovém prostředí NetBeans IDE 8.0.2. Archiv *Exploding-web.zip* obsahuje adresář projektu, který je možné otevřít v NetBeans IDE a přímo spustit (klávesou F6), čímž se při výchozím nastavení prostředí spustí webový prohlížeč s aplikací.

## Softwarové a hardwarové požadavky

Předpokladem je nainstalovaný webový server Apache Tomcat verze alespoň 7.0, na který se připojí webový prohlížeč přes nakonfigurovaný port (standardně 8084). Aplikace by se tedy standardně měla nacházet na adrese *http://localhost:8084/Exploding/*. Případně je možné nasadit Java archiv aplikace *dist/Exploding-web.war* ručně.

Pro umožnění převodu OBJ modelů do formátu JSON a přidruženého BIN, což však není nezbytné pro funkčnost aplikace, je potřebné mít v systémové proměnné *PATH* nastavenou cestu k adresáři, kde se nachází interpret jazyka Python 2 přístupný pod příkazem *python*.

Aplikace vyžaduje grafickou kartu s ovladači podporujícími OpenGL verze alespoň 2.0 a webový prohlížeč s podporou technologie WebGL.

## Databáze a konfigurace

Aplikace nepoužívá databázi. Konfigurační soubory aplikace se nacházejí ve složce *web/WEB-INF/properties/*. Veškeré potřebné knihovny jsou součástí aplikace nebo webového serveru.

# Obsah přiloženého CD

| | |
|---|---|
| *project/* | *Projekt NetBeans se zdrojovým kódem aplikace včetně zkompilované aplikace.* |
| *models/* | *Soubory testovacích 3D modelů.* |
| *text/* | *Text bakalářské práce.* |