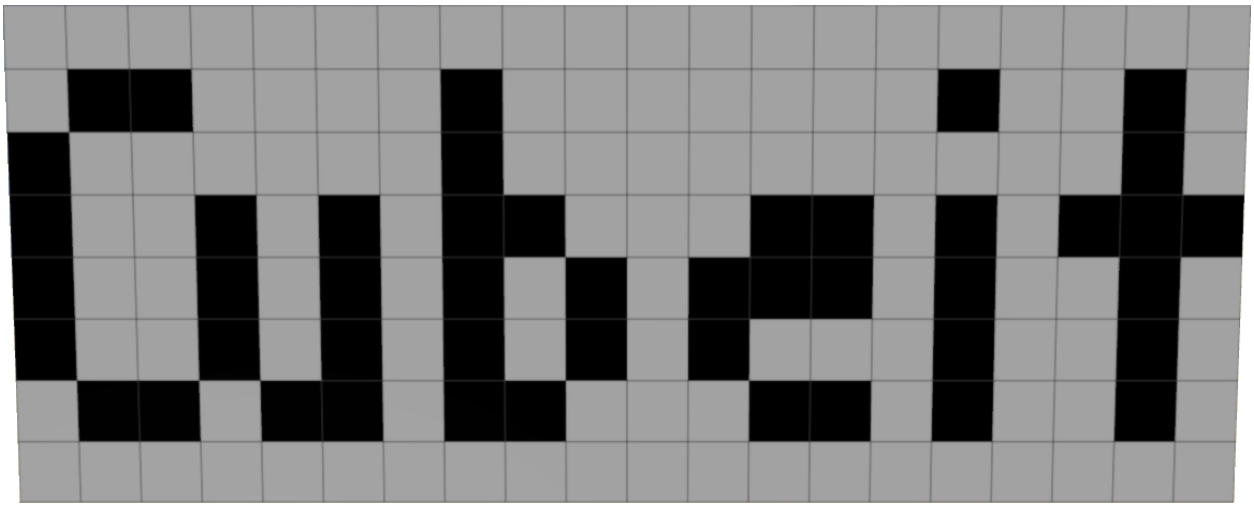
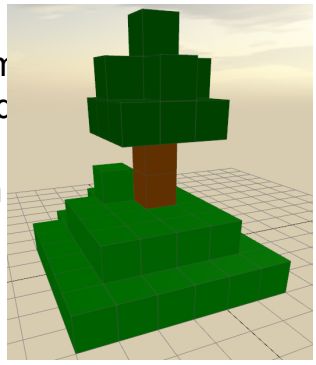


Dokumentation des Voxelpainters Cubeit

In diesem Dokument stelle ich vor was mein Auftrag war, ich zeige wie ich das 3D Zeichnungsprogramm umgesetzt habe und wie man es benutzt. Weiter gebe ich einen kleinen Ausblick was man noch machen könnte und ziehe ein Fazit. Im Anhang findet man was noch zu tun wäre, was meine Quellen waren und wie man eine Entwicklungsumgebung auf dem PC schafft.

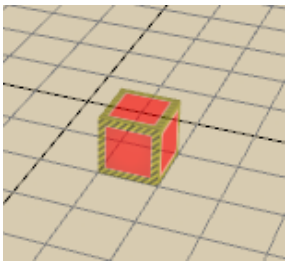
Anforderungen / Use case



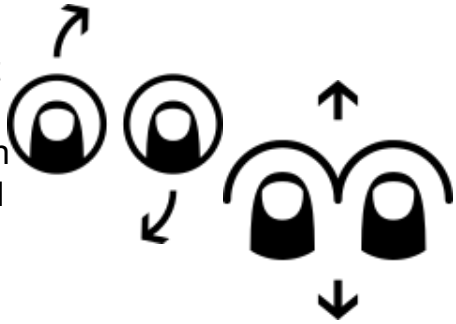
Meine Aufgabe war es ein Tool zu programmieren mit welchem man möglichst einfach Dreidimensionale Objekte zeichnen und als STL-Datei downloaden kann. Damit man beim Zeichnen nicht mit komplizierten Werkzeugen arbeiten muss lassen sich die Objekte wie in Minecraft einfach in einzelnen Blöcken bauen.

Da die Webapplikation vorallem mit Tablets genutzt werden wird, sollen 3 Modi programmiert werden, einen Baumodus, einen Löschmodus und einen Ansichtsmodus. Denn so kann mit gleichen Gesten unterschiedliches Verhalten erzeugt werden.

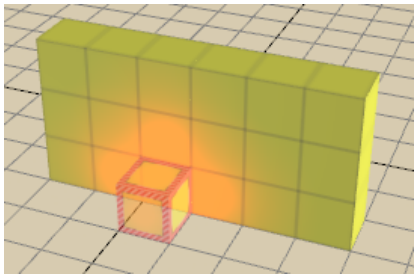
Im Baumodus soll jeweils dort wo sich die Maus befindet oder bei Touchfähigen Geräten wo sich der Finger befindet, ein halb transparenter schwarz-gelb gestreifter Block erscheinen, welcher anzeigt wo gebaut werden soll, diesen bezeichnen wir als Ghostblock. Bei einem Klick soll an der Stelle dieses Ghostblocks ein Block gebaut werden.



Wenn man die Maus bewegt während man die Maustaste gedrückt hält soll man sich um das Objekt drehen. Bei Touchfähigen Geräten soll man sich Horizontal um das Objekt drehen können in dem man zwei Finger sich gegenüber im Kreis bewegt. Vertikal in dem man mit zwei Fingern hoch und runter fährt.

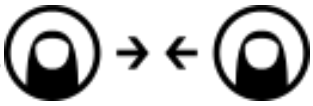


Der Löschmodus soll eigentlich genau gleich sein, nur das der transparente rot-weiss gestreifte Löschblock (Ghostblock) nur erscheint wenn dort auch etwas zu löschen ist, und bei einem Klick soll ein Block gelöscht werden.



Im Ansichtsmodus soll nichts gebaut oder gelöscht werden können und es soll kein Ghostblock erscheinen, dafür soll man sich mit einem Finger um das Objekt drehen können.

In allen Modi soll mit dem Mausrad oder in dem man Zwei Finger auseinander und zusammen bewegt, heran und heraus gezoomt werden können.



Die gezeichneten Objekte sollen gespeichert werden können und sollen wieder abrufbar sein. Damit man nicht immer an das Speichern denken muss, soll man gleich am Anfang aufgefordert werden das Objekt zu benennen, damit das Objekt dann laufend automatisch gespeichert werden kann.
Es soll jeweils eine Info erscheinen wenn gespeichert wurde. Angezeigt soll auch werden wie das Objekt heisst das momentan geöffnet ist.

Die gezeichneten Dreidimensionalen Objekte sollen als STL - Datei gedownloadet werden können. Dabei soll man die Option haben ob hohlräume gefüllt werden sollen oder nicht und ob die Kanten abgerundet werden sollen. Wenn man das heruntergeladene Objekt auf einem 3D Drucker ausdrucken will sind dies beides Möglichkeiten die Druckqualität zu verbessern.

Da die Baufläche begrenzt ist, kann es sein das man auf einer Seite an den Rand kommt und auf der gegenüber liegenden Seite noch Platz wäre. Deshalb soll es eine Zentrierfunktion geben, welche das Objekt in die Mitte der Baufläche verschiebt und es auf sie herunter setzt wenn es vorher schwebte.

Weiter soll es eine Funktion geben welche jeden Block des Objekts neu setzt, und zwar in der Reihenfolge wie ein 3D Drucker vorgehen würde (z.B.Ultimaker) und mit einer Pause zwischen jedem Block.

Für den Fall, dass man zum Beispiel aus versehen ein Block löscht soll es eine Rückgängig Funktion geben.

Obwohl das STL Dateiformat Farben nicht speichert soll man vor verschiedenen Farben zum Bauen auswählen und sie kombinieren können. Falls das STL in Zukunft von einem Format abgelöst wird welches Farbe unterstützt.

Um einem den Einstieg in das Programm zu erleichtern soll es eine Seite geben, welche die wichtigsten Funktionen kurz beschreibt.

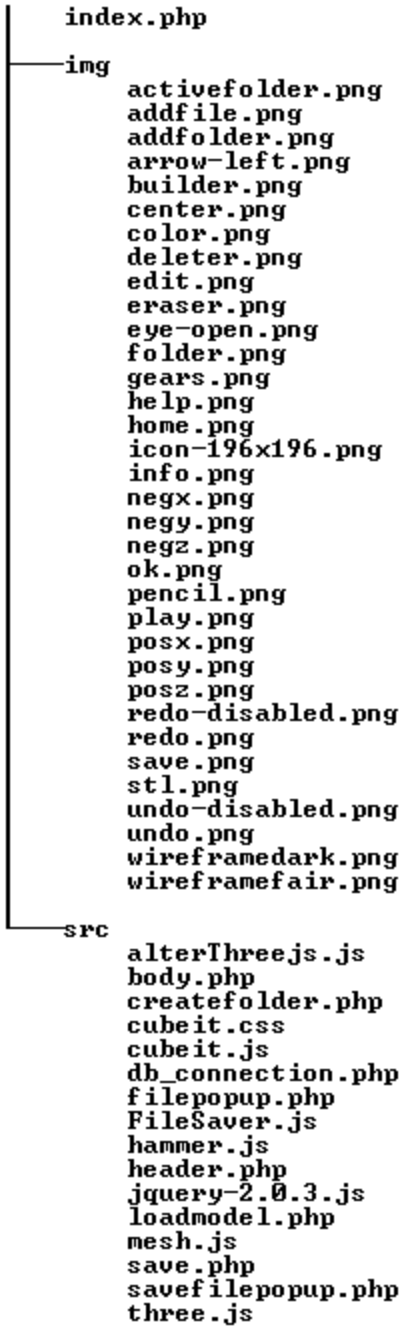
Implementierung

Bevor ich mich an das Tool machen konnte Beschäftigte ich mich mit der Technologie WebGL und kam so auf die JS-Bibliothek THREE.js. Mit dieser Hilfe ist es wesentlich einfacher WebGL zu benutzen. Später kamen dann noch hammer.js für verschiedene Gesten, und jQuery.js für einen abstrahierten Zugang zur AJAX-Kommunikation mit dem Server hinzu.

Dateistruktur

Um den Ordner mit der Webseite etwas aufgeräumt zu haben, habe ich einen Ordner für alle Bilder und Icons “img” und einen Ordner für allen Code “src” gemacht. Einzig das index.php sollte sich nicht da drin befinden.

- In der Datei **index.php** werden andere PHP Dateien hinzugefügt.
- **alterThreejs.js** verändert Shader der THREE.js.
- **body.php** enthält die HTML Struktur der Seite.
- **createfolder.php** speichert den Ordner in der Datenbank.
- **cubeit.css** enthält die Formatierung der HTML Elemente.
- **cubeit.js** beinhaltet die meisten Javascript Funktionen und ist grösstenteils selbst geschrieben.
- **db_connection.php** enthält drei Funktionen für Verbindungen mit der Datenbank.
- **filepopup.php** bereitet den Inhalt für das gleichnamige Popup auf.
- **FileSaver.js** wird benötigt um eine Datei herunter zu laden.
- **hammer.js** wird verwendet um verschiedene Touchevents zu erzeugen.
- **header.php** fügt alle Javascript-Dateien hinzu und enthält den HTML-header.
- **jquery-2.0.3.js** wird für die AJAX abfragen verwendet.
- **loadmodel.php** lädt die Objekte von der Datenbank und sendet sie an die Webseite.
- **mesh.js** generiert die STL Datei
- **save.php** speichert die gezeichneten Objekte samt Bild, und beim automatischen speichern werden sie überschrieben.
- **savefilepopup.php** bereitet den Inhalt für das gleichnamige Popup auf.
- **three.js** enthält umfassende Bibliothek zur Nutzung von WebGL.



Wichtige Funktionen

cubeit.js - centerObject()

Ich beschreibe diese Funktion näher weil ich diese komplett selber geschrieben habe.

Ich fand es wäre praktisch wenn man das Objekt in der Mitte zentrieren könnte, damit man z.B. es wieder besser anschauen kann oder man auf allen Seiten wieder weiter bauen kann. Weiter habe ich dafür gesorgt das bei diesem Prozess das Objekt auch auf die Baufläche zurückgesetzt wird wenn es in der Luft schwebt.

Da ich die Baufläche in Richtung X und Z auf -100 und +100 beschränke und in Richtung Y auf 0 und 200, ist die Mitte bei 0/0/0.

```
function centerObject(){
    var maxx = 100;
    var minx = -100;
    var maxz = 100;
    var minz = -100;
    var maxy = 200;
    var xmove = 0;
    var zmove = 0;
    var ymove = 0;
    for ( var i = 0; i < scene.children.length; i++){
        var object = scene.children[ i ];
        if (object.name == "cube") {
            if (object.position.x<maxx){maxx=object.position.x}
            if (object.position.x>minx){minx=object.position.x}
            if (object.position.z<maxz){maxz=object.position.z}
            if (object.position.z>minz){minz=object.position.z}
            if (object.position.y<maxy){maxy=object.position.y}
        }
    }
    buildinghistoryredo.splice(0, buildinghistoryredo.length);
    if (buildinghistoryundo.length > 0){
        undo[0].style.display = "none";
        undo[1].style.display = "block";
    }else{
        undo[1].style.display = "none";
        undo[0].style.display = "block";
    }
    redo[1].style.display = "none";
    redo[0].style.display = "block";

    xmove = Math.floor((minx+maxx)/20)*10;
    zmove = Math.floor((minz+maxz)/20)*10;
    ymove = maxy-5;
    for (i = 0; i < scene.children.length; i++){
        var object = scene.children[ i ];
        if (object.name == "cube") {
            object.position.x -= xmove;
            object.position.z -= zmove;
            object.position.y -= ymove;
            object.updateMatrix();
        }
    }
    closePopup();
}
```

cubeit.js - generateHash()
Diese Funktion erkläre ich näher weil sie die zum rekonstruieren des Objekts nötigen Daten möglichst kompakt zusammenfasst.

Um die gezeichneten Modelle in der Datenbank zu speichern, verwende ich eine spezielle Komprimierung. Dabei wird durch alle gezeichneten Würfel iteriert und jeweils die Information einem Array angehängt die sich im Vergleich zum letzten Würfel verändert hat. mit encode(array) wird dieses Int array dann in eine Zeichenkette verwandelt und weiter in der Datenbank abgespeichert.

```
function generateHash(){
    var data = [],
    current = { x: 0, y: 0, z: 0, c: 0 },
    last = { x: 0, y: 0, z: 0, c: 0 },
    code;
    for ( var i = 0; i < scene.children.length; i++){
        var object = scene.children[ i ];
        if (object.name == "cube") {
            current.x = ( object.position.x - 5 ) / 10;
            current.y = ( object.position.y - 5 ) / 10;
            current.z = ( object.position.z - 5 ) / 10;
            current.c = colors.indexOf(object.material.color.getHex());
            code = 0;
            var torem = 0;
            if ( current.x != last.x ){code += 1;}
            if ( current.y != last.y ){code += 2;}
            if ( current.z != last.z ){code += 4;}
            if ( current.c != last.c ){
                code += 8;
                if ( current.c%2 == 1) {
                    torem = 1;
                    code += 16;
                }
            }
            data.push(code);
            if ( current.c != last.c ) {
                data.push( (current.c-torem)/2 );
                last.c = current.c;
            }
            if ( current.z != last.z ) {
                data.push( current.z + 10 );
                last.z = current.z;
            }
            if ( current.y != last.y ) {
                data.push( current.y );
                last.y = current.y;
            }
            if ( current.x != last.x ) {
                data.push( current.x + 10 );
                last.x = current.x;
            }
        }
    }
    return encode( data );
}
```

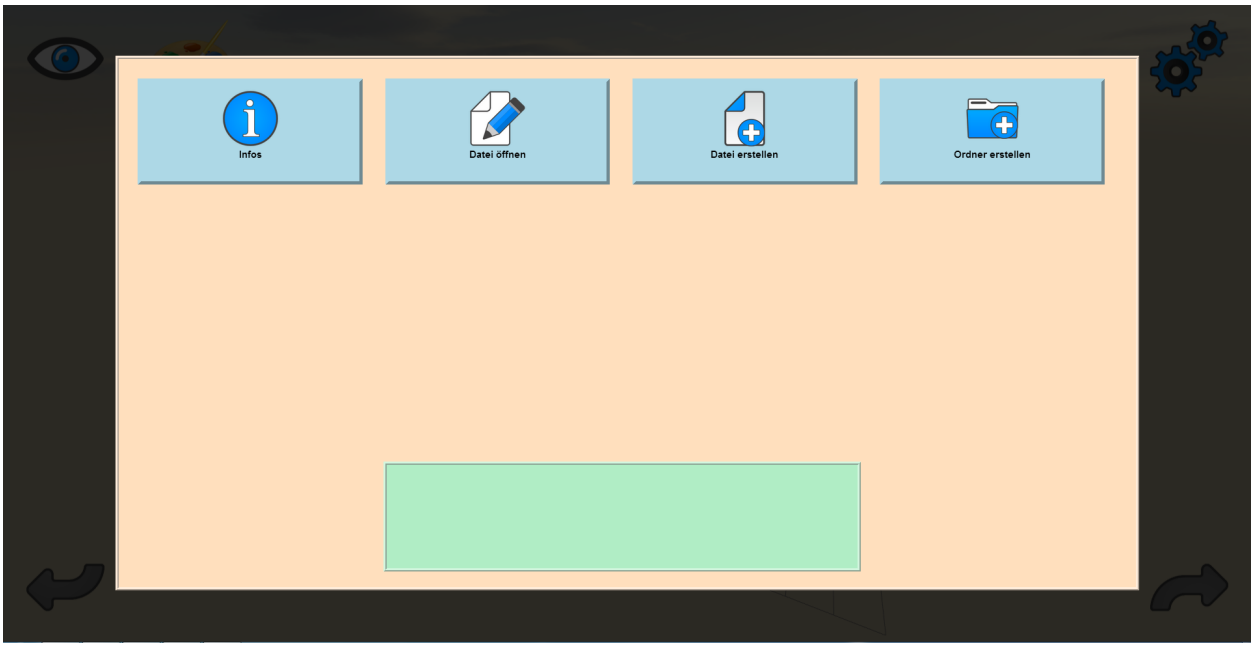
save.php
Dieses Script habe ich ausgewählt weil es sowohl zum speichern als auch zum aktualisieren dient.

Um ein gezeichnetes Objekt in der Datenbank zu speichern habe ich dieses PHP-Skript geschrieben. Es überprüft ob alle benötigten Werte vorhanden sind und erstellt oder aktualisiert die werte in der Datenbank

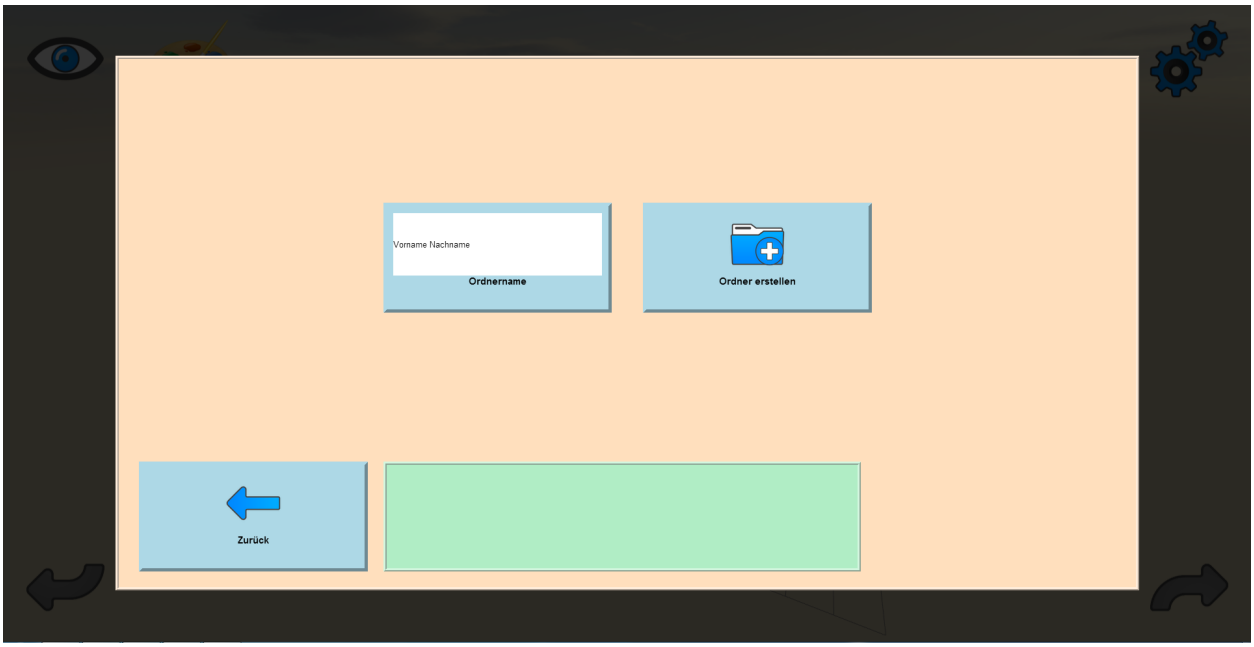
```php
<?php
    if(!isset($_POST["action"])) die();
    if(!isset($_POST["data"])) die();
    if(!isset($_POST["hash"])) die();

    $data = substr($_POST["data"], strpos($_POST["data"], ",")+1);
    $hash = mysql_real_escape_string($_POST["hash"]);

    include 'db_connection.php';

    openDBConnection();

        if($_POST["action"] == "normal"){
            executeQuery("INSERT INTO object (Objectname, Data, Folder_id, Image)
                VALUES ('".$_POST["filename"]."', '".$hash."',
                ".$_POST["folderid"].", '".$data."')");

            echo executeQuery("select Id_object From object
                ORDER BY Id_object DESC Limit 1")[0][0];
        }
        else if($_POST["action"] == "auto"){
            echo executeQuery("update object SET Data = '".$hash."',
                Image = '".$data."' Where Id_object = ".$_POST["modelid"]);
        }

    closeDBConnection();
?>
```

von der Idee zur fertigen Zeichnung

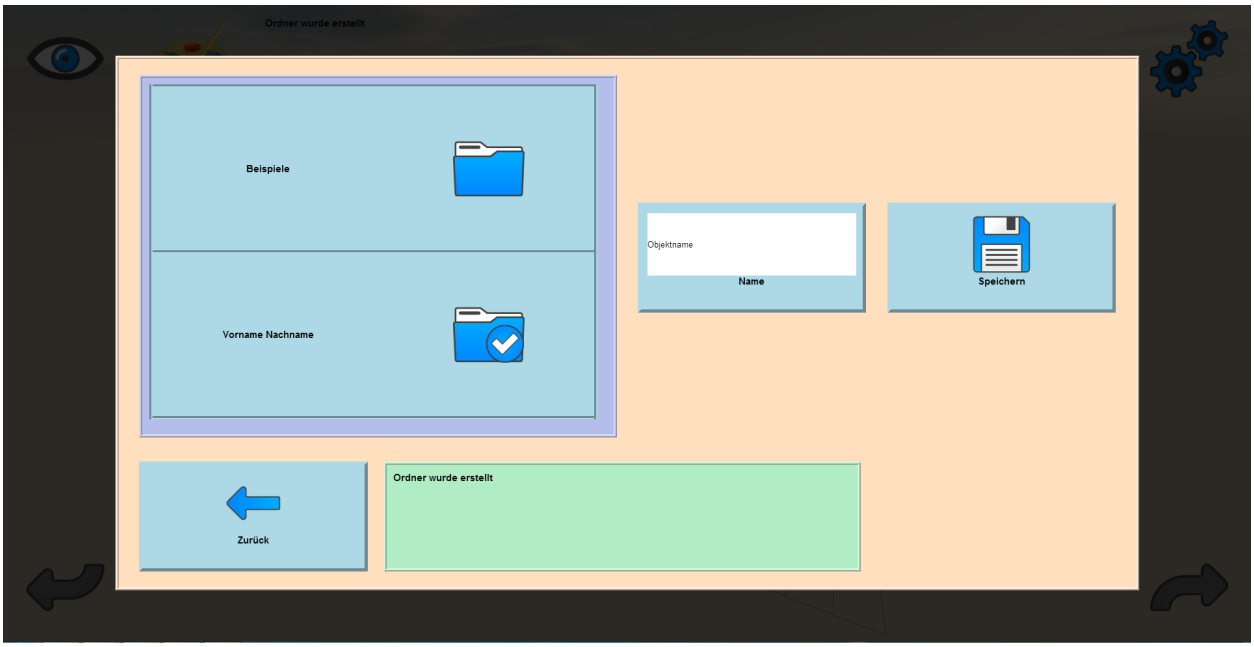
Wenn man die Webapplikation öffnet sollte es ungefähr so aussehen wie auf dem Bild unten zu sehen ist. Als erster Schritt empfehle ich den Ordner erstellen Knopf zu drücken um einen neuen Ordner für die Zeichnungen zu erstellen.



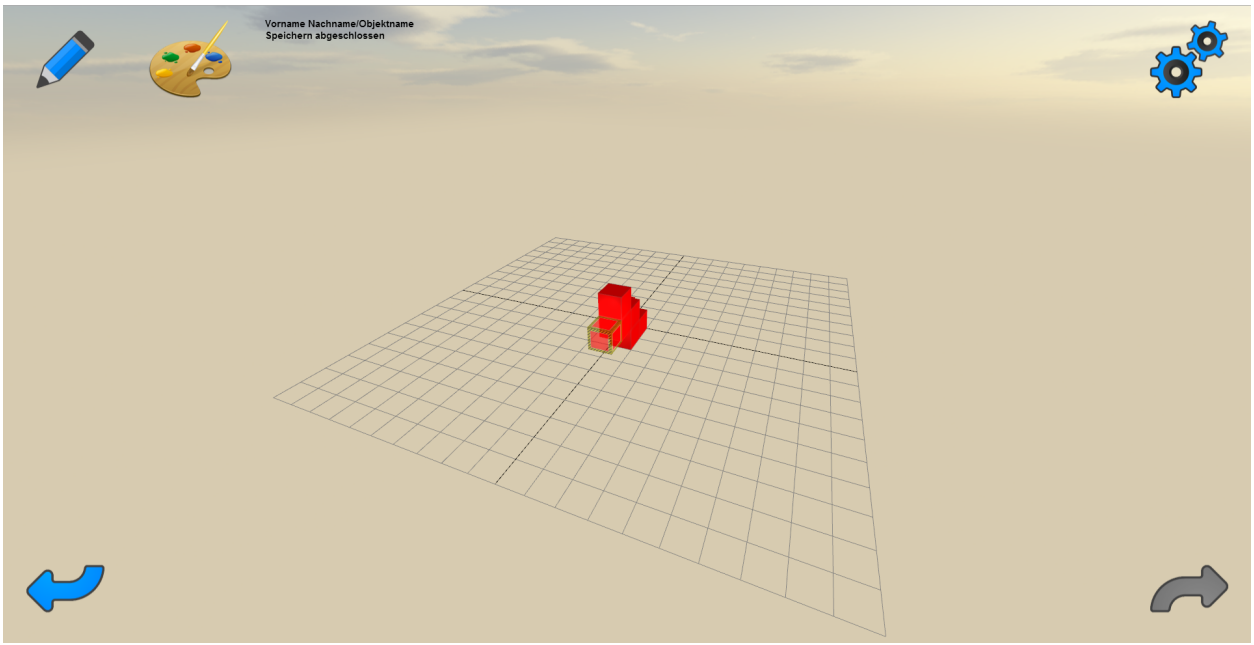
Als Namen für den Ordner kann man zum Beispiel den eigenen Namen verwenden.



Weiter geht es damit etwas zu zeichnen, dazu muss man auf Datei erstellen klicken. Dann kann man links den gerade erstellten Ordner auswählen und rechts den Namen eingeben welcher die Zeichnung haben soll. Um mit zeichnen zu beginnen kann man speichern drücken, wenn man abbrechen möchte zurück.

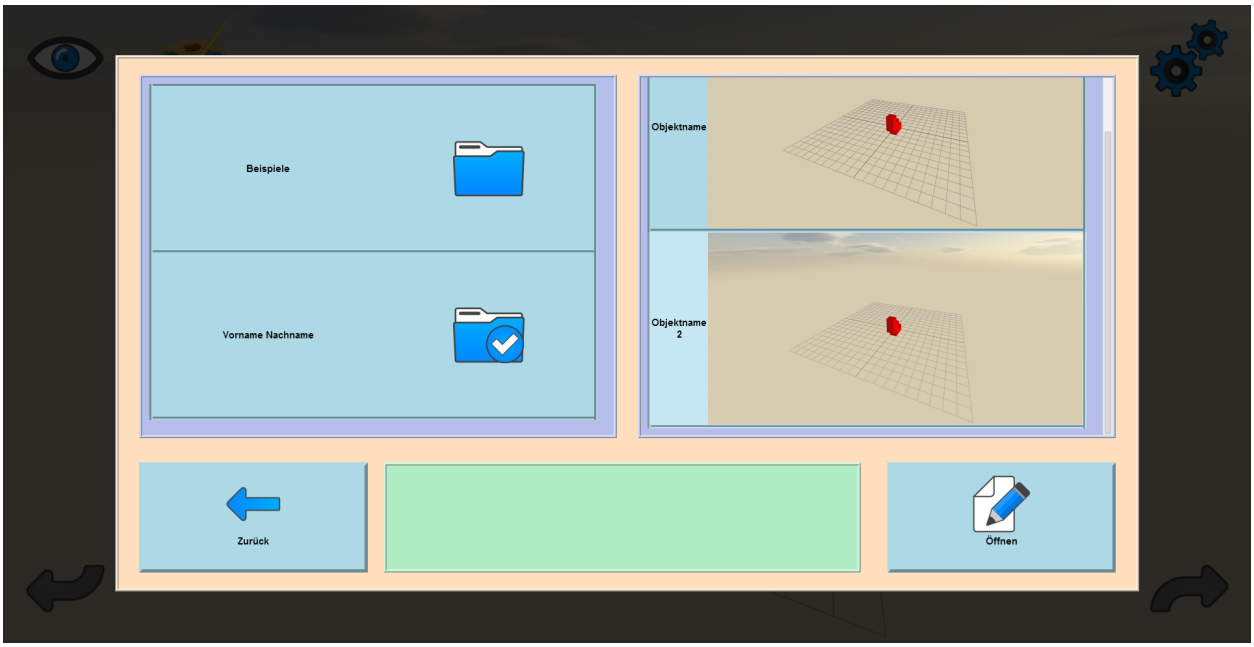


Wenn man Speichern gedrückt hat sieht man nun oben in der Mitte den Ordner und Zeichnungsnamen und kann mit zeichnen beginnen. Wenn man in der linken oberen Ecke auf die Farbpalette klickt kann man die Farbe auswählen mit welcher man zeichnen möchte.

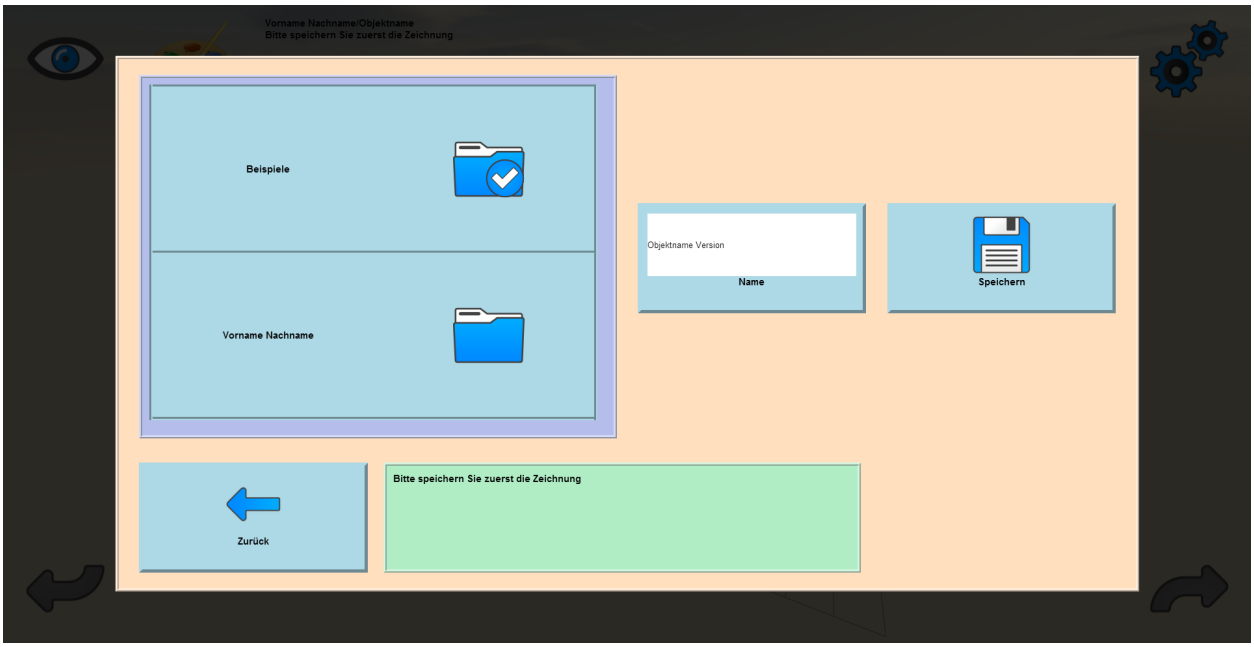


Wenn man mit Zeichnen fertig ist, oder man keine Zeit mehr hat kann man die Seite Bedenkenlos verlassen.

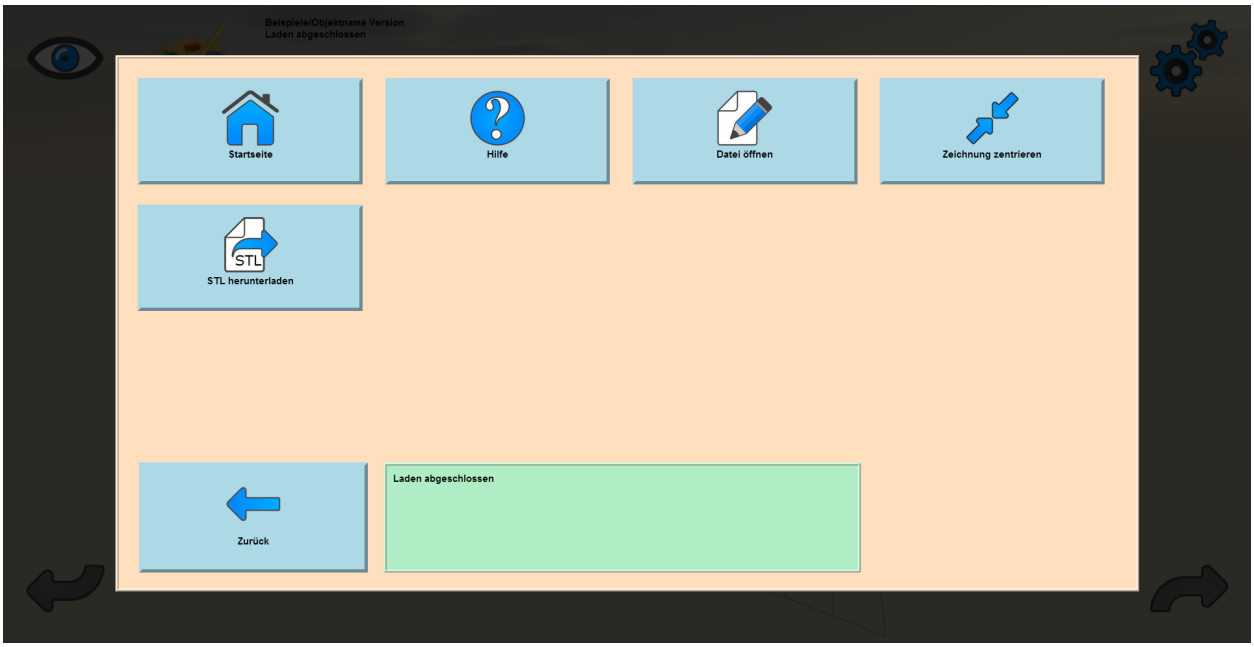
Wenn man das nächste mal wieder kommt kann man unter Datei öffnen links den Ordner auswählen und rechts dann die Zeichnung die man ansehen oder bearbeiten möchte. Zum Schluss noch unten rechts öffnen anklicken und man kann die Zeichnung von allen Seiten betrachten.



Damit man nun aber auch weiterzeichnen kann wird man beim wechseln in den Bau oder Löschmodus aufgefordert die Zeichnung wieder zu speichern. Dies ist nötig damit man nicht die Zeichnungen von andern bearbeiten kann.
Wenn die Zeichnung wieder gespeichert wurde kann man wieder weiter daran arbeiten.



Um die Zeichnung zu Zentrieren und wenn sie schwebt um sie auf die Baufläche zurück zu setzten kann man unter Optionen Zentrieren auswählen.



Weiter gibt es die Möglichkeit die Zeichnung als STL zu exportieren. Auch diese Funktion findet man unter Optionen.

Ausblick

Um die Bedienung zu verbessern wäre es noch gut wenn man im Bau und im Löschmodus mit zwei Finger hoch und runter bewegen könnte. Jedoch kann man die Bedienung auch komplett überdenken und alle Modi zusammenfügen und den Ghostblock in der Mitte zwischen zwei Finger positionieren und die Kamerabewegung auf einem Finger lassen. Löschen könnte man dann in dem man lange den Finger auf das Tablett legt oder mit der Maus rechtsklickt.

Beim exportieren des STL wären die Optionen zum füllen von Hohlräumen und glätten von kanten noch von vorteil. Weiter könnte man die Drucksimulation, welche mit der Funktion simulatePrint bereits begonnen wurde noch fertig machen. Zum zentrieren des Objekts wäre es noch von Vorteil, wenn der Benutzer darauf hingewiesen wird das dies nicht rückgängig gemacht werden kann, oder dass es möglich gemacht wird diesen Schritt rückgängig zu machen.

Im Popup “Hilfe” könnte man eine kleine Hilfe einbauen welche dem Benutzer die ersten Schritte erklären würde. Im Popup “Info” fände ich es wichtig wenn man die Quellen angibt und ein Link zur dazu gehörenden Lizenz.

Fazit

Auch wenn ich nicht alles erreicht habe was Anfangs das Ziel war bin ich mit der Arbeit zufrieden. Es gäbe zwar noch viel was ich noch ändern würde, aber leider reicht die Zeit dazu nicht mehr.

Ich denke ich konnte mit diesem Programm viel lernen und da das dass eigentliche Ziel war habe ich dieses voll und ganz erreicht.

Anhang

ToDo

Das Userinterface sollte noch ansprechender und benutzerfreundlicher gestaltet und auch auf Tablets optimiert werden. Es gibt zum Beispiel das Problem das alles verzogen wird wenn sich auf dem Tablet die Tastatur öffnet. Weiter sollte der Benutzer häufiger mit der Funktion setMessage() über das momentane Geschehen informiert werden. Zum Beispiel “STL generieren ist zu xx% abgeschlossen”. Die Funktion setMessage wiederum sollte dahingehend verbessert werden, das sie die Information auch wieder verschwinden lässt.

Quellen

Für den Hintergrund habe ich eine Skybox gemacht. Das Bildmaterial dazu habe ich von hier:
http://reije081.home.xs4all.nl/skyboxes/
Lizenz: http://creativecommons.org/licenses/by-nc-sa/3.0/
Ich habe die heruntergeladene Skybox lediglich zerschnitten um sie in sechs Stücken zu haben.

Die Buttons habe ich von iconfinder von einem Iconset:
https://www.iconfinder.com/search/?q=iconset%3Asnipicons
Lizenz: http://creativecommons.org/licenses/by-nc/3.0/
Ich habe ein paar neue Icons aus diesen zusammengesetzt.

Weiter habe ich diverse JavaScript Bibliotheken und Codeteile verwendet:
Three.js:
http://threejs.org/
Hammer.js:
http://eightmedia.github.io/hammer.js/
JQuery.js:
http://jquery.com/
FileSaver.js:
https://github.com/eligrey/FileSaver.js/blob/master/FileSaver.js

Ich konnte viel von den Fragen und Antworten von Stack Overflow profitieren:
http://stackoverflow.com/

Wie Installiere ich es auf einem PC (zur Weiterentwicklung)

Um die Webapp auf ihrem lokalen PC zu installieren müssen sie zuerst inmal XAMPP installieren. Dazu besuchen sie am besten diese Seite:
http://www.chip.de/downloads/XAMPP_22023279.html
und laden XAMPP dort herunter. Dann installieren sie XAMPP, bei denn Komponenten brauchen sie nur Apache, MySQL, PHP und phpMyAdmin aus zu wählen. Als Verzeichnis empfehle ich C:\xampp zu lassen. Eventuell müssen sie bei Skype unter Aktionen → Optionen → Erweitert → Verbindung das Häkchen bei "Ports 80 und 443 als Alternative …" heraus nehmen. Nun können sie das cubeit.zip unter C:\xampp\htdocs entpacken. Weiter müssen sie nun das XAMPP Control Panel öffnen um Apache und MySQL zu starten, eventuell müssen sie das Control Panel dazu als Administrator öffnen. Nun können sie einen Browser starten und oben localhost eingeben und "enter" drücken, so verbinden sie auf den lokalen Apache dienst. Klicken sie nun unter Tools auf phpMyAdmin, und dort oben auf SQL. Öffnen sie nun die Datei cubeit.sql, welche sie in ihrem Ordner Webapplikation_cubeit unter Datenbank finden. Kopieren sie den kompletten Inhalt in die Textbox welche sie zuvor im Browser bei phpMyAdmin öffneten, und klicken sie auf "ok". Wenn sie nun auf localhost/cubeit verbinden sollte die Startseite des Tools angezeigt werden. Getestet wurde mit XAMPP 1.8.3, PHP 5.5.6, MySQL 5.6.14 und Google Chrome 32.