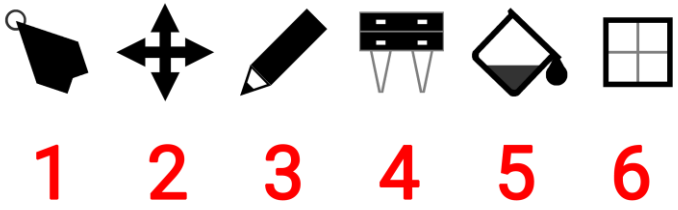
PAccem

Einleitung

Tab-Leiste

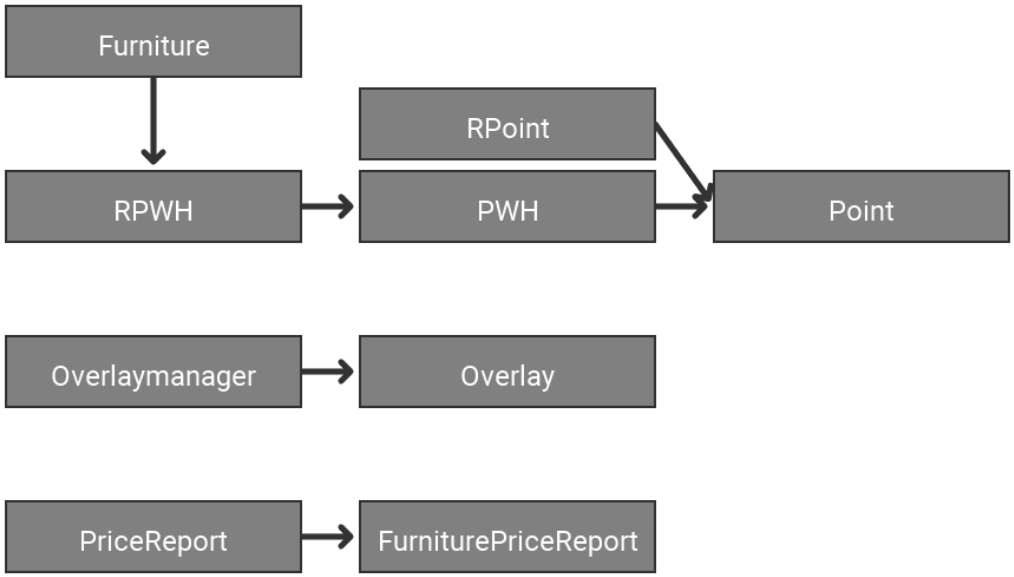
Nummer	Aktion
1	Öffnet ein Popup, um einen neuen Raum mit auswählbarer Größe zu erstellen
2	Ändert die Ansicht zwischen 2D und 3D
3	Öffnet die Raumliste mit allen Räumen in data/rooms
4	Öffnet das Speicher Menu um den aktuellen Raum abzuspeichern
5	Öffnet die Raumgruppenliste
6	Öffnet das Preis Menu, welches die Kosten für den aktuellen Raum nennt.
7	Öffnet das Einstellungmenu
8	Öffnet das Zurücksetzen Popup, um den aktuellen raum zurückzusetzen.
9	Öffnet das Info Popup. (Version, Autor, Githublink)

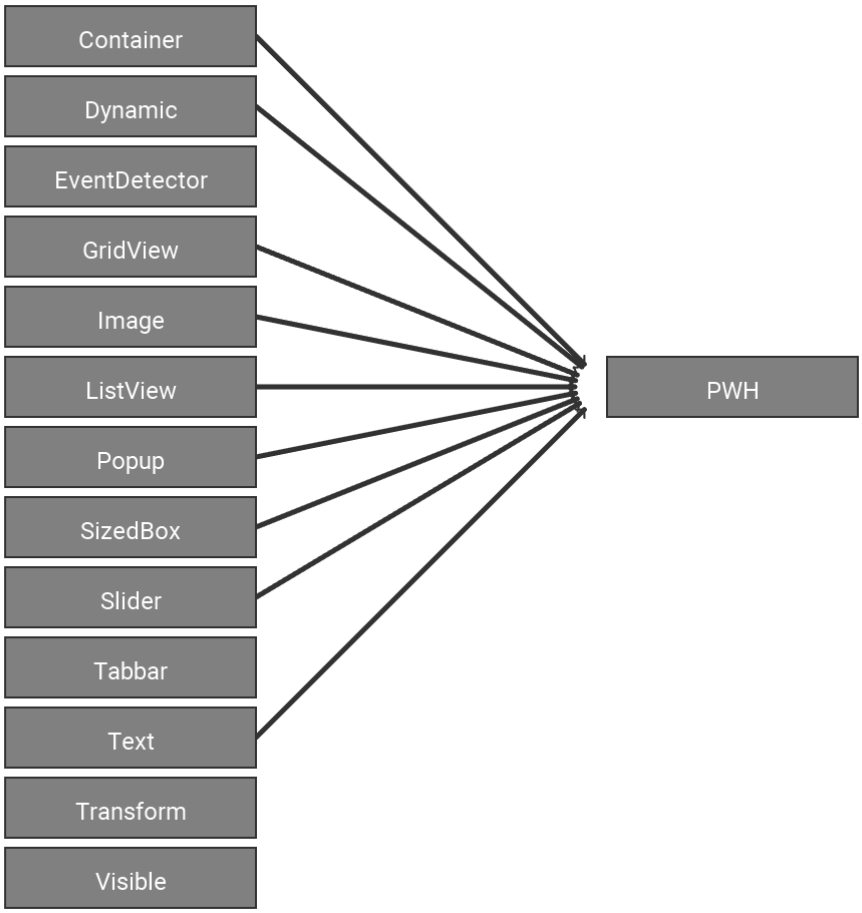
Werkzeug-Leiste



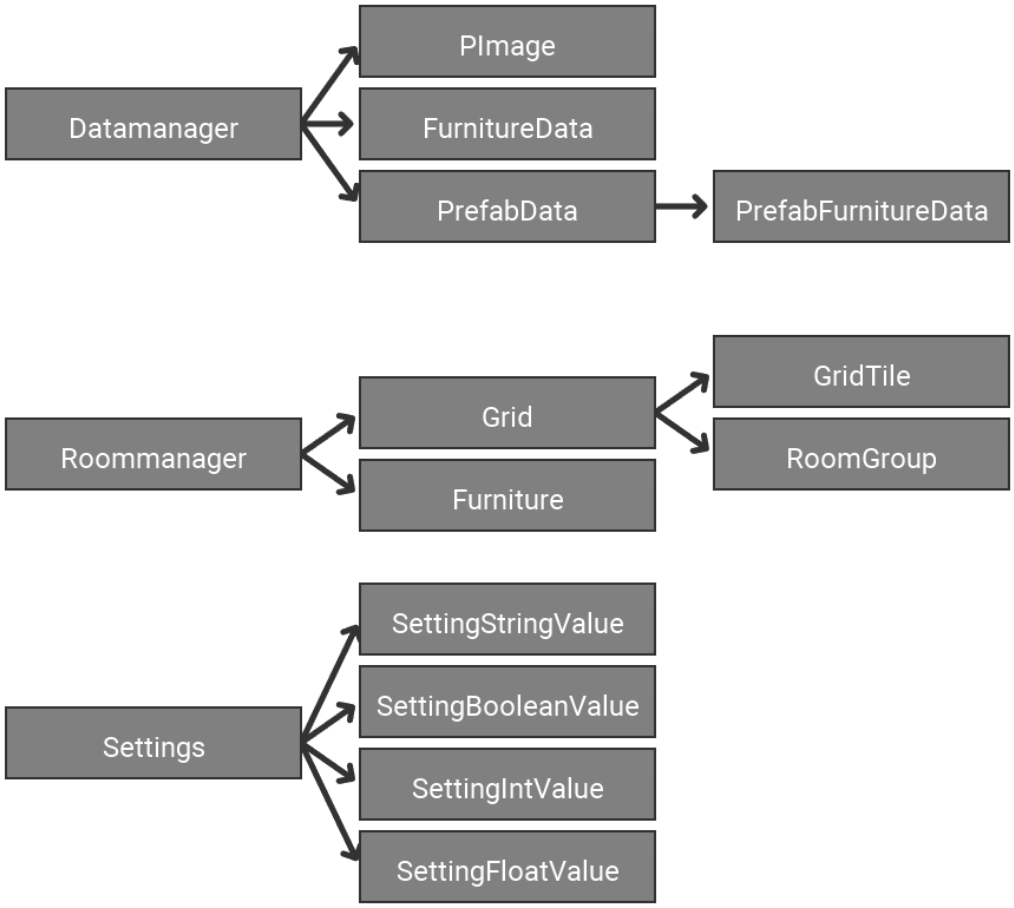
Nummer	Werkzeug, Verwendung
1	Bewegen Werkzeug, zum Umsehen in der 2D Ansicht
2	Zeichnen Werkzeug, zum Raumgrundriss erstellen
3	Möbel Werkzeug, zum Platzieren von neuen Möbeln
4	Auswahl Werkzeug, zum Verschieben oder löschen von Möbelstücken
5	Füll Werkzeug, zum Ausfüllen eines Bereiches im Raumgitter
6	Fenster Werkzeug, zum Platzieren von Fenstern

Vererbung


Furniture
RPoint
RPWH
PWH
Point
Overlaymanager
Overlay
PriceReport
FurniturePriceReport


Container
Dynamic
EventDetector
GridView
Image
ListView
Popup
SizedBox
Slider
Tabbar
Text
Transform
Visible
PWH

Komposition



Einstellungen

Name	Beschreibung
Standartraumname	Name des Raumes, welcher geladen wird, wenn das Programm startet
Sprache	Sprache des Programms
Schriftart	Schriftart name
Dark mode	Verdunkelt das Programmaussehen
Verstecke Overlay	Versteckt das Overlay
Vollbild	Vollbildmodus
Nutze OpenGl Renderer	Ermöglich die 3D Ansicht (benötigt eine OpenGL Grafikkarte und x64)
Breite	Breite des Programmfensters
Höhe	Höhe des Programmfensters
Kantenglättung	Menge an Kantenglättung (2 -> 2x Kantenglättung, etc.)
Gitterliniendicke	Dicke der Gitterlinien

Klassenübersicht

PAccem.pde/PApplet

PApplet ist eine Klasse in welcher sowohl eine main (setup), als auch eine loop(draw) Funktion enthalten ist. Sie repräsentiert die höchste Klasse für den Programmierer und definiert den Startpunkt. Jegliche Maus und Tastatur Events werden in dieser Klasse übergegeben und ggf. verarbeitet.

Weitere Information über PApplet und das Processingkonstrukt: siehe Anhang.

Variablen

ApplicationManager am: Verwaltet Programmelemente, wie Titel, Fenstergröße und Programmstart

Settings st: Lädt und speichert alle im Programm enthaltenen Einstellungen ab.

LanguageManager lg: Lädt die aktuelle Sprachdatei und gibt deren Daten wieder. (siehe: data/assets/lang/)

RoomManager rm: Verwaltet den gesamten Raum(Gitter, Möbel, Nutzereingaben)

DataManager dm: Verwaltet alle Daten, wie 3D Modelle, Bilder, usw.

OverlayManager ov: Enthält das GUI

Clipper cl: Ermöglicht das Pushen und Poppen von clip() (siehe: Processing push(), pop(), clip())

PGraphics pg: Grafikoberfläche für 3D-Grafik

PShader blurshader: Ein Shader welcher Gaussian Blur Effekt enthält. (siehe: data/assets/shader/blur.glsl)

PFont font: Die aktuelle Schriftart

boolean usegl: ob die usegl Einstellung zum Programmstartzeitpunkt an oder aus war.

boolean allowcgol: ?

ArrayList toovmessages: enthält alle Nachrichten, die an die Konsole gesendet werden sollen. (siehe: Overlay)

int[] c: Ein Array aus Farbwerten welche sich nach dem Dunkelmodus ausrichten. (0-8 => 0 - 255 oder 255 - 0)

boolean isKeyUp, isKeyRight, isKeyLeft, isKeyDown, isKeyT: Status der einzelnen Tasten

boolean deb: Debugmode

boolean disableblur: ob Weichzeichnen deaktiviert ist

Funktionen

void settings(): Wird ausgeführt bevor das Programmfenster erstellt wird.

void setup(): Wird ausgeführt, nachdem das Programmfenster erstellt wurde.

void draw(): Wird für jedes Bild ausgeführt.

void mouseWheel(MouseEvent e): Wird ausgeführt, wenn der Nutzer sein Mausrad dreht.

void mouseDragged(): Wird ausgeführt, wenn der Nutzer seine Maus bewegt, wenn eine Maustaste gedrückt ist.

void mouseReleased(): Wird ausgeführt, wenn der Nutzer eine Maustaste loslässt.

void mousePressed(): Wird ausgeführt, wenn der Nutzer eine Maustaste drückt.

void keyPressed(KeyEvent e): Wird ausgeführt, wenn der Nutzer eine Tastaturtaste drückt.

void keyReleased(): Wird ausgeführt, wenn der Nutzer eine Tastaturtaste loslässt.

ApplicationManager

ApplicationManager initialinziert (alle) Variablen in PAccem.pde/PApplet und erstellt anhand der Programmfenstereinstellungen das Programmfenster. Ebenso werden Aktionen wie Argumente, Farbmanagement, Schriftart, Programmtitel und Programmfenster verwaltet.

Variablen

String setfontrawinput: Wird vom Thread in setFontRaw() als Parameter genutzt

Funktionen

void initSettings(): Wird ausgeführt bevor das Programmfenster erstellt wird

void initSetup(): Wird ausgeführt, nachdem das Programmfenster erstellt wurde

void setTitle(String name): Legt den Programmfenstertitel fest

void setFont(String newfontname): Legt die aktuelle Schriftart fest. Es wird ermittelt ob es sich bei der gewählten Schriftart um eine aus der Roboto Schriftfamilie handelt und wenn dies nicht der Fall ist, wird setFontRaw() als Thread ausgeführt, um die Schriftart zu ermitteln.

void setFontRaw(): Wird von setfont(String newfontname) als Thread ausgeführt.

void recalculateColor(): Legt die Farbwerte in PAccem/PApplet gegeben nach dem Dunkelmodus fest

void manageArgs(): Verarbeitet alle Argumente, welche an das Programm übergeben wurden.

Name	Aktion
-debug	Aktiviert den Debugmode
-nofilter	Deaktiviert Filter

void loop(): Es wird nachgesehen, ob sich die Programmfenstergröße geändert hat und daraufhin die Fenstergrößeneinstellungen angepasst und ggf. die Größe der 3D-Grafikoberfläche(pg) angepasst.

Clipper

Clipper ermöglich es zu die clip Einstellung zu "pushen" und zu "poppen". Dies benötig die clip() Funktion aus Processing, welche den Bereich in der eine Zeichenfunktion zeichnet, eingrenzt.

Variables

ArrayList clips: Enthält die aktuelle und alle vergangenen clip Einstellungen zu welchen man "poppen" kann.

Functions

Clip get(): gibt die aktuelle clip Einstellung.

void pushClip(): "pusht" die aktuelle clip Einstellung. Das bedeutet, dass die aktuelle clip Einstellung gespeichert wird und auf die gegebene Einstellung umgestellt wird.

void popClip(): "poppt" die aktuelle clip Einstellung. Das bedeutet, dass die ehemalige clip Einstellung, welche durch pushClip() gespeichert wurde, wieder verwendet wird und somit die aktuelle überschrieben wird.

Extra

Clip: Enthält eine clip Einstellung, also einen Bildschirmbereich

DataManager

DataManager lädt (alle) Assets aus data/assets, welcher er abspeichert, validiert und ggf. in ein leichter zugängliches Format verwandet. (siehe: FurnitureData, PrefabData)

Variablen

final PImage[] icons: Liste aller Icons

final PImage[] extras: Liste aller extra Bilder

final FurnitureData[] furnitures: Liste aller Möbel welche der Nutzer verwenden kann.

final PrefabData[] prefabs: Liste aller Fertigteile welcher der Nutzer verwenden kann.

final PShader[] filters: Liste aller Filter/Shader.

Funktionen

int[] validate(): Sieht nach, ob alle Möbel in ihren Fertigteilen ins Fertigteil hineinpassen.

boolean validateId(int id): Sieht nach, ob ein Möbelstück mit der gegebenen id existiert.

FurnitureData getFurnitureData(int id): gibt die Möbelinformationen mit der gegebenen id.

PrefabData getPrefabData(int id): gibt die Fertigteilinformationen mit der gegebenen id.

Extra

PImage

PImage enthält ein Bild.

PShape

PShape enthält ein 3D Model.

FurnitureData

FurnitureData enthält alle Informationen über ein Möbelstück.

PrefabFurnitureData

PrefabFurnitureData enthält alle Informationen über ein Möbelstück in einem Fertigteil.

PrefabData

PrefabData enthält alle Informationen über ein Fertigteil.

Furniture

Die Furniture Klasse repräsentiert ein einzelnes Möbelstück, welches von der RoomManager Klasse verwaltet wird.

Variablen

int id: id des Möbelstücks

int price: Preis des Möbelstücks

color tint: Färbung des Möbelstücks

Funktionen

void draw(boolean viewmode, boolean selected): Zeichnet/Rendert das Möbelstück

void drawFrame(boolean selected): Zeichnet/Rendert eine Box auf dem Möbelstücks

boolean checkover(): Ermittelt, ob die Maus auf das Möbelstück zeigt

boolean checkover(int xpos, int ypos): Ermittelt, ob das Möbelstück in der gegebenen Gitterposition liegt

Clip getBoundary(): gibt die Grenzen des Möbelstücks zurück

boolean setXPos(int value): Legt die x Position des Möbelstücks fest

boolean setYPos(int value): Legt die y Position des Möbelstücks fest

void move(int dx, int dy): Bewegt das Möbestückt

Grid

Die Grid Klasse ist für das dem Raumplaner zugrundeliegendem Gitter zuständig.

Variablen

GridTile[][] tiles: 2 dimensionales Gitter

ArrayList roomgroups: Liste aller Raumgruppen

Funktionen

void draw(boolean viewmode, float gts): Zeichnet/Rendert das Gitter

void fillTool(boolean value, int x, int y): Wendet das Füll Werkzeug an

boolean setTileState(boolean value, int x, int y): Legt den Status des gegebenen

Kachels fest boolean getTileState(int x, int y): Gibt den Status des gegebenen Kachels zurück

boolean setTile(GridTile value, int x, int y): Legt die Variablen des gegebenen Kachels fest

GridTile getTile(int x, int y): Gibt die Variablen des gegebenen Kachels zurück

boolean isinGrid(int x, int y): Ermittelt, ob die gegebene Position im Gitter liegt

boolean isRoomGroupinuse(int id): Ermittelt, ob eine gegebene Raumgruppe im Gitter verwendet wird

void removeRoomGroup(int id): Entfernt eine gegebene Raumgruppe aus der Liste

void cgol(): hmmmm?

int getActiveTiles(): Gibt die Anzahl an Kacheln an, welche teil des Raums sind.

Extra

GridTile

Enthält alle Informationen über ein einzelnes Kachel.

RoomGroup

Enthält alle Informationen über eine Raumgruppe.

LanguageManager

LanguageManager lädt eine Sprachdatei (siehe: data/assets/lang/) und gibt deren Werte aus

Variablen

JSONObject data: Aktuelle Sprachdaten

Funktionen

boolean setLang(String newlang): Legt die aktuelle Sprache fest

String get(String key): Gibt die Übersetzung mit dem gegebenen Schlüsselwort zurück

OverlayManager

Die OverlayManager Klasse ist für das gesamt GUI zuständig. Die Variablen und die build() Funktion können beliebig verändert werden, um jedes mögliche GUI zu erstellen. Sie ist eine Erweiterung der Overlay Klasse, welche Aufgaben wie Zeichnen/Rendern und Event Handling übernimmt.

Variablen

final int xoff: Wird verwendet, um das Gitter am Overlay auszurichten

final int yoff: Wird verwendet, um das Gitter am Overlay auszurichten

boolean drawpopup: Sichtbarkeitsstatus des Popups

int tabid: wird von Tabbar verwendet (siehe: OTabbar.pde)

String newroomname: Der Name für einen neuen Raum

int newroomxsize, newroomysize: Die Größe für einen neuen Raum

Object tempdata: temporäre Variable mit verschiedenen Verwendungen (meisten zum transfer von Daten mit dem Popup)

ArrayList messages: Alle Nachrichten welche in der Nachrichten Box sind

int consoleoff: Offset der Nachrichten (scrollen)

boolean drawconsole: Sichtbarkeitsstatus der Nachrichten Box

final int messageboxheight: Höhe der Nachrichten Box

Funktionen

void build(): Erstellt das Overlay

void checkMessages(): Fügt alle Nachrichten in toovmessages (siehe: PAccem/PApplet) der Nachrichten Box hinzu

void printMessage(String text): Fügt eine Nachricht der Nachrichten Box hinzu

void drawPopup(int id): Öffnet ein Popup (unterschiedlich je nach id)

ID	Popup
0	Benötig Neustart
1	Neuer Raum
2	Infos
3	Zurücksetzen
4	Entferne Raumgruppe
5	Neue Raumgruppe
6	Wähle Farbe
7	Aktiviere CGOL
8	Standartraum überschreiben?
9	Benötigt openGL Renderer

int getXOff(): gibt die xoff Variable des OverlayManagers, je nach dem, ob das Overlay angezeigt wird.

int getYOff(): gibt die yoff Variable des OverlayManagers, je nach dem, ob das Overlay angezeigt wird.

Overlay

Das Overlay basiert auf einem Parent Child Konzept in dem ein Element ein oder mehrere Kinder/Elemente beinhalten und ein Kind/Element nur ein Parent besitzt. Z.B eine ListView kann eine Serie an Container enthalten, welche wiederum Text oder Image enthalten. Dies ähneld dem Aufbau eines Baumdiagramms. Die Implementierung Idee basiert auf der von Google entwickelten SDK namens Flutter.

Variablen

Object[] items: Liste aller Elemente im Overlay

boolean visible: Sichtbarkeitsstatus des Overlays

Funktionen

void setitems(Object[] items): Nimmt alle OverlayElemente an sich und positioniert sie am Nullpunk.

void draw(): Zeichnet/Rendert das Overlay

boolean isHit(): Ermittelt, ob die Maus auf dem Overlay liegt

void mouseWheel(MouseEvent e): Wird ausgeführt, wenn der Nutzer sein Mausrad dreht

boolean mousePressed(): Wird ausgeführt, wenn der Nutzer eine Maustaste drückt und gibt zurück, ob der Click etwas im Overlay ausgelöst hat

void mouseReleased(): Wird ausgeführt, wenn der Nutzer eine Maustaste loslässt

boolean mouseDragged(): Wird ausgeführt, wenn der Nutzer seine Maus bewegt und gibt zurück, ob die Bewegung etwas im Overlay ausgelöst hat

void keyPressed(KeyEvent e): Wird ausgeführt, wenn der Nutzer eine Tastaturtaste drückt

void keyReleased(): Wird ausgeführt, wenn der Nutzer eine Tastaturtaste loslässt

RoomManager

Die RoomManager Klasse ist die wichtigste Klasse im Programm, welcher die eigentlichen Raumplaner Funktionen implementiert.

Variablen

ArrayList furnitures: Liste aller Möbel im Raum

Grid roomgrid: Das aktuelle Raumgitter

int selectionid: der Index des aktuell ausgewählten Möbelstücks (-1 = kein)

String name: Name des Raums

float xoff, yoff, scale: Variablen für die 2D Ansicht

float dxoff, dyoff, dzoff, angle1, angle2, dspeed: Variablen für die 3D Ansicht

int gridtilesize: Größe eines Kachels

int tool: id des aktuell gewählten Werkzeuges

ID	Werkzeug
0	Bewegen

1 Zeichnen

2 Möbel oder Fertigteil platzieren

3 Möbelstück auswählen

4 Füllen

5 Fenster platzieren

boolean viewmode: Wahr = 3D Ansicht, Falsch = 2D Ansicht

ArrayList dragtiles: Liste aller Kachel über welche der Nutzer bereits gezeichnet hat.

boolean dragstate: Zeichenstatus

int newfurnitureid: id von neu Platzierten Möbelstücken

int newroomgroup: id der aktuell ausgewählten Raumgruppe zum Zeichnen

boolean isprefab: ob gerade ein Fertigteil platziert wird

color furnituretint: Färbung von neu Platzierten Möbelstücken

Funktionen

void mouseWheel(MouseEvent e): Wird ausgeführt, wenn der Nutzer sein Mausrad dreht.

void mouseDragged(): Wird ausgeführt, wenn der Nutzer seine Maus bewegt.

void mouseReleased(): Wird ausgeführt, wenn der Nutzer eine Maustaste loslässt.

void mousePressed(): Wird ausgeführt, wenn der Nutzer eine Maustaste drückt.

void keyPressed(KeyEvent e): Wird ausgeführt, wenn der Nutzer eine Tastaturtaste drückt.

void keyReleased(): Wird ausgeführt, wenn der Nutzer eine Tastaturtaste loslässt.

float getXPos(): Verwandelt die Maus X Position in eine Raumgitter X Position.

float getYPos(): Verwandelt die Maus Y Position in eine Raumgitter Y Position.

boolean isFurniture(int xpos, int ypos): Gibt an, ob sich ein Möbelstück an der gegebenen Position gibt

boolean isFurnitureBlock(boolean debug): Gibt an, ob sich ein Möbelstück an der Maus Position platzierbar ist

int getXGridSize(): Gibt die X Größe/Breite des Raumgitters.

int getYGridSize(): Gibt die Y Größe/Höhe des Raumgitters.

String[] loadRooms(): Gibt eine Liste aller Räume (in data/rooms/)

void save(String name): Speichert den aktuellen Raum (in data/rooms/) mit dem gegebenen Namen

void load(String name): Lädt den gegebenen Raum (in data/rooms/)

int getPriceReport(): Gibt einen Preisbericht zurück, welche Informationen über die Raumkosten enthält.

void reset(): Setzt den Raummanager zurück

void newRoom(int xsize, int ysize): Erstellt einen neuen Raum mit der gegebenen Größe

void switchViewmode(): änder die Ansicht (2D -> 3D, 3D -> 2D)

void resetCamera(boolean viewmode): setzt die gegebene Kameraansicht zurück

void draw(): Zeichnet/Rendert den Raum

Extra

PriceReport

Enthält einen gesamten Kostenbericht über den aktuellen Raum.

FurniturePriceReport

Enthält einen Kostenbericht alle Möbel des aktuellen Raums.

Settings

Die Settings Klasse lädt alle Einstellungen (aus data/settings.json), welcher er abspeichert, validiert und in ein leichter zugängliches Format verwandet. (siehe: Variablen)

Variablen

final SettingStringValue[] strings: Einstellung vom Typ String/Text

final SettingBooleanValue[] booleans: Einstellung vom Typ Boolean/Wahrheitswert

final SettingIntValue[] ints: Einstellung vom Typ Int/ganze Zahl

final SettingFloatValue[] floats: Einstellung vom Typ Float/Kommazahl

Funktionen

int getSize(): Ermittelt die gesamte Anzahl an Einstellungen

String set(int id, String value): Setzt die gegebene Einstellung zum gegebenen Wert. (automatische Datentypumwandlung)

SettingValue get(int id): Gibt die Einstellung mit der gegebenen id

void load(): Lädt die Einstellungen von data/settings.json, wenn möglich

void save(): Speichert die Einstellung in data/settings.json

Extra

SettingStringValue

Ein Einstellungtyp, welcher einen String/Text abspeichert.

SettingBooleanValue

Ein Einstellungtyp, welcher einen Boolean/Wahrheitswert abspeichert.

SettingIntValue

Ein Einstellungtyp, welcher einen Boolean/Wahrheitswert abspeichert. Der Wert kann durch einen Minimal- Maximalwert eingegrenzt werden.

SettingFloatValue

Ein Einstellungtyp, welcher einen Float/Kommazahl abspeichert. Der Wert kann durch einen Minimal- Maximalwert eingegrenzt werden.

SettingValue

Eine als Rückgabewert verwendete Klasse, um einen Wert von einem beliebigen Datentyp auszugeben.

Extra

Baseclasses

In BaseClasses.pde werden Klassen zur Vererbung definiert und "temporäre" Klassen, welche in abstrakten Funktionen verwendet werden um Konstante Werte weiter zu geben.

Point: 2D Punkt

PWH extends Point: Punkt, Breite und Höhe

RPoint extends Point: Rotation und Punkt

RPWH extends PWH: Rotation, Punkt, Breite und Höhe

class Temp: Speichert einen Integer/ganze Zahl

class STemp: Speichert einen String/Text

Basefunctions

In BaseFunctions.pde werden vielseitige Funktionen definiert.

String getAbout(): Gibt den Info Text

void setKey(int k, boolean bool): Legt den Status von manchen Tasten fest. (Pfeiltasten, T)

String cap(String str): Verwandelt den ersten Buchstaben eines Strings/Texts in Großschrift

String fixLength(String str, int length, char c): Füllt einen String/Text mit einem gegebenen Zeichen bis der String/Text eine gegebene Länge erreicht hat.

void printColor(int c): Schreibt eine Farbe in die Konsole.

void printColorhex(int c): Schreibt eine Farbe in Hexadecimal in die Konsole.

Constants

In basefunctions.pde sind Konstanten definiert.

final String appname: Name des Programms

final String appversion: Version des Programms

final String appmaker: Name des Autors

final String githublink: Link zum GitHub Repository

Alle weiteren im Programm verwendeten Konstanten sind in PConstants definiert. (siehe Anhang)

Overlay Elemente

CheckBox

Ein CheckBox for displaying boolean settings.

Container

Enthält ein Kind/Element.

Wenn man mit der Maus auf den Container zeigt wird die Farbe verändert.

Dynamic

Erstellt anhand der abstrakten Funktion getItem() ein Kind/Element welches dann angezeigt wird.

EventDetector

Erkennt Maus und Keyboard Events vom Nutzer auf sein Kind/Element.

Filter

Wendet einen Filter/Shader auf sein Kind/Element an.

GridView

Ähnlich wie ListView, jedoch können mehrere Kinder pro Reihe angezeigt werden.

Image

Zeigt ein Bild an.

ListView

Eine einfache Liste aus Elementen/Kindern. Die Liste kann in jede Richtung zeigen.

Popup

Zeigt ein Kind/Element im Bildschirmmittelpunkt an. Der Hintergrund wird verdunkelt und weichgezeichnet.

Weichzeichner: siehe data/assets/shader/blur.glsl

SizedBox

Wird in ListView verwendet um Leerstellen zu erstellen. Der Expand Parameter verursacht, dass die SizedBox die maximale Größe in der ListView annimmt, dabei wird er sich den Platz mit anderen SizedBoxen fair teilen müssen. :(

Slider

Ein horizontaler Schieberegler.

Tabbar

Enthält eine ListView als Tab Leiste und zeigt je nach dem welcher Tab ausgewählt ist das dazugehörige Kind/Element an.

Text

Zeigt einen unveränderbaren Text an.

SetValueText

Zeigt einen vom Nutzer veränderbaren Text an.

SetValueStyle

Definiert das Verhalten von SetValueText.

int type: Datentyp der Eingabe

ID	Datentyp
0	String/Text
1	Boolean/Wahrheitswert

2 Integer/ganze Zahl

3 Float/Kommazahl

int maxlength: die maximale Eingabenlänge

GetValueText

Zeigt einen unveränderbaren Text an, welcher durch eine abstrakte Funktion ermittelt wird.

Transform

Verschiebt sein Kind/Element und oder positioniert es in einer Bildschirmecke.

Visible

Versteckt sein Kind/Element anhand seiner abstrakten Funktion

Extra

OBase: Enthält Klassen, Enums, Interfaces und Funktionen welche vom gesamten Overlay verwendet werden.

Box: Wird im Overlay verwendet, um die Grenzen des Kindes/Elements weiterzugeben.

TabData: Enthält Informationen, welches verwendet wird, um die Tabs zu erzeugen.

Builder: Erstellt einen Array aus Objects(Kinder/Elemente) mithilfe einer abstrakten Funktion.

ListViewBuilder: Tut das gleiche, jedoch werden die Objects danach in eine ListView verwandelt.

Dir: Gibt eine Richtung an. (siehe ListView Code)

Align: Gibt eine Position an. (siehe Align Code)

Fit: Gibt das Verhalten von Image an. (siehe Image Code)

IOverlay: Enthält Funktionen, welche alle Overlay Elemente enthalten müssen.

Dynamic casting: Da Java keine dynamische Variable besitzt, wird hier mithilfe des "instanceof" operators die Klasse ermittelt und der Befehl nach dem Casting für das Object ausgeführt.

Anhang

PAccem Github Repository: Link

Processing Internetseite: Link

Processing Github Repository: Link

PApplet Dokumentation: Link

PConstants Dokumentation: Link