--------Video bis Teil 11-----------

Hinzufügen von Klasse „ScreenSize“ in „core“

Speichert Bildschirmbreite / höhe

Erstellt variable „ground“, für den Boden auf dem sich der Character bewegen soll

Getter-Methoden für alles

In class GameObject.java

Setzt y wert von new position auf ground – 64 -> character spawnt 64p über Boden

In class Motion.java

Ausklammern von „isRequestingDown“

Hinzufügen boolean „falling“

Hinzufügen von if-Abfrage

*//wenn Position 64p (Character-Größe = 64, deswegen 128) über Boden wird fallling true  
//wenn Position größer als Boden und nicht Up requestet wird wird falling true*

Hinzufügen von if-Abfrage

*//wenn Position kleiner-gleich Boden wird falling false*

Hinzufügen von if-Abfrage

Wenn falling = true wird zu deltaY 1 addiert -> player sinkt um 2.5p

Zu if(controller.isRequestingUp()) wird && !falling hinzugefügt -> wenn fällt kann nicht nochmal in der Luft springen

Ausprobieren was beste spring-speed

Feststellen dass sich nichts ändert, egal welche zahl zu deltaY hinzuaddiert wird

Ausklammern von vector.normalize() -> Problem behoben

Feststellen dass deltaY = 2 zu wenig und = 3 zu viel

deltaX & deltaY zu doubles machen

requstingUp/falling werden zu 2.7

requstingLeft/requestingRight werden zu 1.5

In class ScreenSize

Hinzufügen von Left & Right border (damit character nicht aus screen rausrennt)

leftBorder = breite / 4 -> linkes viertel bleibt frei

rightBorder = breite / 3 \* 2 -> rechtes drittel bleobt frei

getter-Methoden dafür

in class Motion.java

hinzufügen in if-Abfrage isRequestingLeft: && Position.*getX*() > ScreenSize.*getLeftBorder*()

hinzufügen in if-Abfrage isRequestingRight: && Position.*getX*() < ScreenSize.*getRightBorder*()

In class GameObject.java

Setzt x wert von new position auf leftBorder -> spawnt an linkem rand

Hinzufügen von sitzender Mikasa in „stand.png“

Class Motion.java

Public static boolean down

Wieder einbeziehen von isRequestedDown -> deltaY += 0.1 (damit ubgedatet), down = true

Alle anderen movements: down = false

Getter für down

Enum Direction

Hinzufügen von D (down)

If(Motion.getDown()) -> D zurückgeben

Creating von class Menu in package menu

String playerName aus Launcher verschieben

Ändern zu string array, momentan nur ein Name gespeichert, final gemacht

Private final JFrame „menu“ mit tite „Menu“

private final int width & height

Erstellen Konstruktor mit int height & width

This.width = width etc.

Menu.setSize auf Hälfte von height & width (da später: height = screenSize.getHeight(), etc.)

menu.setDefaultCloseOperation(EXIT\_ON\_CLOSE);

menu.getContentPane().setBackground(Color.CYAN);

menu.setResizable(false);

menu.setLayout(null);

menu.setLocationRelativeTo(null);

Erstellen von JButton b1 mit text „Start Game“

* 1/3 der Breite/höhe von JFrame
* Position: 1/3 der Breite/höhe von JFrame, damit mittig

Menu.setVivible(true);

Hinzufügen von actionlistener

* Startet game wenn butten gedrückt

Launcher

game start entfernen

new Menu(ScreenSize.getWidth(), ScreenSize.getHeight());

Menu.class

JButton Array mit der Länge aller Character Names

private final JButton[] buttons = new JButton[playerName.length];

unter b1 declaration:

schleife: erstellt für jeden angegebenen character einen button

mit Größe: halb so breit wie fenster, 1/12 so hoch wie fenster -> in breite genug spielraum für lange namen

position: ¼ des fensters von linken rand entfernt, 1/5 des Fensters von oberem Rand entfernt, ab dann immer 5px unter darüber liegenden Button

neuer ActionListener: setzt Game.setName auf ausgewählten namen, schleife: entfernt jeden button, repaintet menu, addet b1

„makeNameNice“ Methode -> Formatiert lowerCase-String mit Unterstrich statt Leertaste zu String mit first Letter after Sace zu Upper Case, Unterstrich zu Leertaste

* Statt „name“ als text in Button, jetzt: makeNameNice(name), damit folder name format bleiben kann, aber auf Buton schön aussieht

actionListener durch lambda ersetzt:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

-> wird zu ->

Ein Bild, das Text, Screenshot, Bildschirm enthält.

Automatisch generierte Beschreibung

Zwischenergebnis: läuft, nur ein Problem:

**Folder loading funktioniert auf Laptop, aber auf PC nicht!**

Nächste Ziele:

* Überschrift über Player-Auswahl
* Zurück-Button in Game
* Block-Entity
* Soll von Player als „second ground“ angesehen werden
* Einlesen von txt-Datei als map

Bsp:

------------------------------------------

----------------------XX------------------

----------------XX------------------------

------------------------------------------

XXXXXXXXXXXXXXXXXXXXXXXXX

- = Luft/nichts; X = Block

SpriteLibrary.java

getFolderNames gelöscht

String[] folderNames = Menu.playerName;

Do ordner so heißen wie player

getSheetsInFolder gibt vorher deklarierten String[] zurück, mit manuell eingetippten file-names (stand.png & walk.png)

* Dieser deklariert in Menu, bei playerNames
* Jetzt funktioniert auch auf PC

Vorbereitung für themes: PATH\_TO\_UNITS = "/sprites/units/" + Menu.getGameTheme()

Menu.java

public static final String[] gameTheme = new String[] {"attack\_on\_titan"};

* Aktuell nur ein Theme

private static int theme;

* Sagt welches Theme verwendet wird

Getter für game theme:

Im Moment: theme = 0, da noch keine game theme auswahl

Return gameTheme[theme];

Game.java

Umbenennen des fenster-titles in "Jump'n'Run - "+ Menu.makeNameNice(Menu.getGameTheme()) +" styled"

Menu.java

Umbenennen von b1 zu startGame

Umbenennen von buttons zu players

Neues JButton Array „themes“ -> länge von gameTheme

Hinzufügen von Color Array bgColor -> für jedes Theme eine Farbe

* Aktuell nur Farbe für AoT
* Getter gibt bgColor(theme) zurück

Hinzufügen von Button “backThemes” & “backPlayers“

* Zurück zu Themes
* Zurück zu Players
* Maße: 76\*10p, es sei denn width/20\*height/20, dann dies, damit Button im Verhältnis groß genug, aber Text immer zu lesen auch wenn display sehr klein

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Jeder neue Button kriegt eigenen getActionListener mit entsprechenden Parametern

Player adding in eigene Methode ausgelagert und backThemes Button geaddet -> wenn Players aufgelistet, Button erscheint, der zurück zu Themes

+ Farbänderung zu bgColor des ausgewählten themes

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Neue Methode add Themes (basically das gleiche wie addPlayers, nur mit Themes und ohne back button) Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

actionListeners von Themes & Players

* Speichert Eingabe, Löscht alles Angezeigte, fügt Folgendes hinzu, updatet Display

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

getTheme:

int theme wird nicht mehr automatisch auf 0 gesetzt

actionListener der Back Buttons:

* Aktuell angezeigtes wird gelöscht, display geupdatet und vorheriges hinzugefügt

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

GameDisplay.java:

* Initialisierung: private JButton back;
* Diesmal fix-Maße (min Maße von Menu)

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Wichtig: setFocusable(false), sonst liegt Fokus auf Button und nicht auf KeyListener -> pfeiltasten funktionieren nicht mehr

ActionListener:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* Öffnet Menu, schließt GameDisplay
* Führt zu Error in public void render(Game game) bei back Button benutzung:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* Try{}catch block darum:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

**Zwischenergebnis:**

Menu:





//back Buttons funktionieren Einwandfrei



//wenn Back Button in GameDisplay benutzt und danch Game erneut gestartet wird (per Menu), funktioniert Springen nicht mehr richtig

Fehler eingegrenzt: liegt an boolean falling in Class Motion, da wenn falling dauerhaft auf false, funktioniert springen (abgesehen von fallen, aber das ja logisch)

Motion.java

* Fehlersuche erfolglos, aber andere Verbesserungsidee:
* Springen nicht linear, sondern exponentiell (realistischer)
* Neue Methode getFallSpeed

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

x = höhe (pixel)

d = Geschwindigkeit (pixel/update)

neue Variable: private double gravity;

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Jeden frame per second, dass up requested wird, wird gravity um 0.5 größer, wenn fällt, dann 0.5 niedriger jeden frame per second, und Y Bewegung wird der Funktion entprechend ausgeführt

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Motion.java

* Speed final gemacht

Game.java

* l. 39 lambda „gameObject -> gameObject.update()” zu method reference “GameObject::update”