

An abstract graphic on the left side of the slide, consisting of a network of white lines and small circles on a blue gradient background, resembling a circuit board or a neural network.

MULTIPLAYER-PROJEKT

CT-PRÄSENTATION VON DOMINIK KAISER UND DAVID BALSACQ 17.5.20

INHALT

- Verlauf des Projekts
 - Planung
 - Vorbereitung
 - Programmieren
- Probleme und Lösungen
- Wurden die Ziele erreicht?
- Demo
- Quellen

DIE PLANUNG

- Planung: Was wollen wir überhaupt machen?
- Vorschläge enthielten: Mensch Ärgere dich nicht, Computerversion des Verrückten Labyrinths, Multiplayer-Pacman-Version...
- Welche Engine?
- Welche Programmiersprache?

ERGEBNISSE DER PLANUNG

- Das waren die Ziele: Multiplayer-Spiel, 2D top-down, es soll Spaß machen, übersichtlich sein, jeder kann es schnell lernen, ohne großen Aufwand spielbar.
- Godot als Engine für das Projekt
- Programmiersprache Godot Script (pythonbasierte Sprache)
- Spieleidee: Zwei Spieler, einer schickt Monster los, der andere verteidigt sich.

VORBEREITUNG

- Eine Menge Youtube-Tutorials
- Godots eigene Tutorials und Website
- Godot-Dokumentation
- Ähnliche Projekte anschauen
- Arbeitsaufteilung
- Austausch und Versionsverwaltung

ERGEBNISSE DER VORBEREITUNG:

- „Das könnte eventuell etwas sehr viel Arbeit werden“
- „Wer soll jetzt nochmal was machen?“
- „Kannst du den Teil machen? Du hast dich doch schon in dem Bereich eingelesen“
- „GitHub funktioniert irgendwie nicht“

PROGRAMMIERUNG

- Das Spiel muss Controller und Tastatur Input erkennen
- Das Spiel muss den Input richtig umsetzen (Steuerung)
- Es müssen Gegner erschaffen werden
- Gegner müssen gespawnt werden
- Gegner müssen sich unterschiedlich verhalten
- Es müssen Wände erschaffen werden
- Kollision muss eingebaut werden
- Spieler muss Angreifen können
- Spieler muss sich bewegen können

DAS ERGEBNISS DER PROGRAMMIERUNG

Scene Project Debug Editor Help

2D 3D Script AssetLib

GLE33

Scene Import

Filter nodes

- TitleScreen
 - Cursor
 - Sprite
 - Area2D
 - CollisionShape2D
 - Start_Game
 - CollisionShape
 - Sprite
 - bg

FileSystem

res://World.gd

Search files

- Switch Sides.tscn
- Sword.gd
- Sword.tscn
- SwordCollision.gd
- Time.gd
- TitleScreen.tscn
- tolles gesicht.png
- tolles schwert.png
- ui_bg.png
- World.gd
- World.tscn
- X-Button.png
- Y-Button.png
- yes.png

Filter scripts

- bullet.gd
- Cursor.gd
- GameMaster.gd
- mobspawner.gd(*)
- PlayerCharacter.gd
- playershot.gd
- RandomMob.gd
- Shooting mob.gd
- standardmob.gd
- Sword.gd
- Time.gd
- World.gd

mobspawner.gd(*)

Filter methods

- _init
- _ready
- _physics_process
- give_dir
- on_hit
- give_knockdir
- knockmobback
- die
- shoot

File Search Edit Go To Debug

Online Docs Request Docs Search Help

```
29 >|
30
31
32 >| func _physics_process(delta):
33 >|
34 >| if attack_cooldown.is_stopped():
35 >| >| shoot()
36 >| >| attack_cooldown.start()
37 >|
38 >| if position.x < 16:
39 >| >| set_position(Vector2(16, position.y))
40 >| >| dir = -give_dir()
41 >| if position.x > WINDOW_WIDTH - 16:
42 >| >| set_position(Vector2(WINDOW_WIDTH - 16, position.y))
43 >| >| dir = -give_dir()
44 >| if position.y < 16:
45 >| >| set_position(Vector2(position.x, 16))
46 >| >| dir = -give_dir()
47 >| if position.y > WINDOW_HEIGHT - 16:
48 >| >| set_position(Vector2(position.x, WINDOW_HEIGHT - 16))
49 >| >| dir = -give_dir()
50
51 >| if knockedback == true:
52 >| >| knockmobback()
53 >| else:
54 >| >| if wait <= 0:
55 >| >| >| randomNr = randi()%360+1
56
57 >| >| >| dir = give_dir()
58 >| >| >| move_and_slide(dir * SPEED)
59 >| >| >| wait = 60
60 >| >| >| #print(randomNr)
61 >| >| else:
62 >| >| >| move_and_slide(dir * SPEED)
63 >| >| >| wait = wait-1
64 >|
65 >| if get_slide_count() > 0:
66 >| >| var collision = get_slide_collision(get_slide_count()-1)
```

Inspector Node

World.gd

Filter properties

- Resource
- Resource
- Reference

4 (114, 19)

Output Debugger Search Results Audio Animation

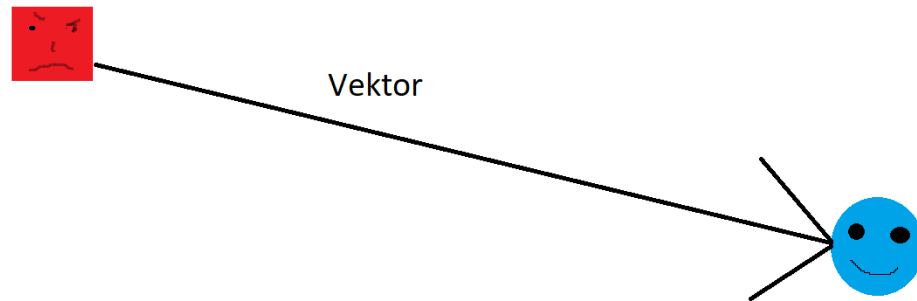
3.2.1.stable



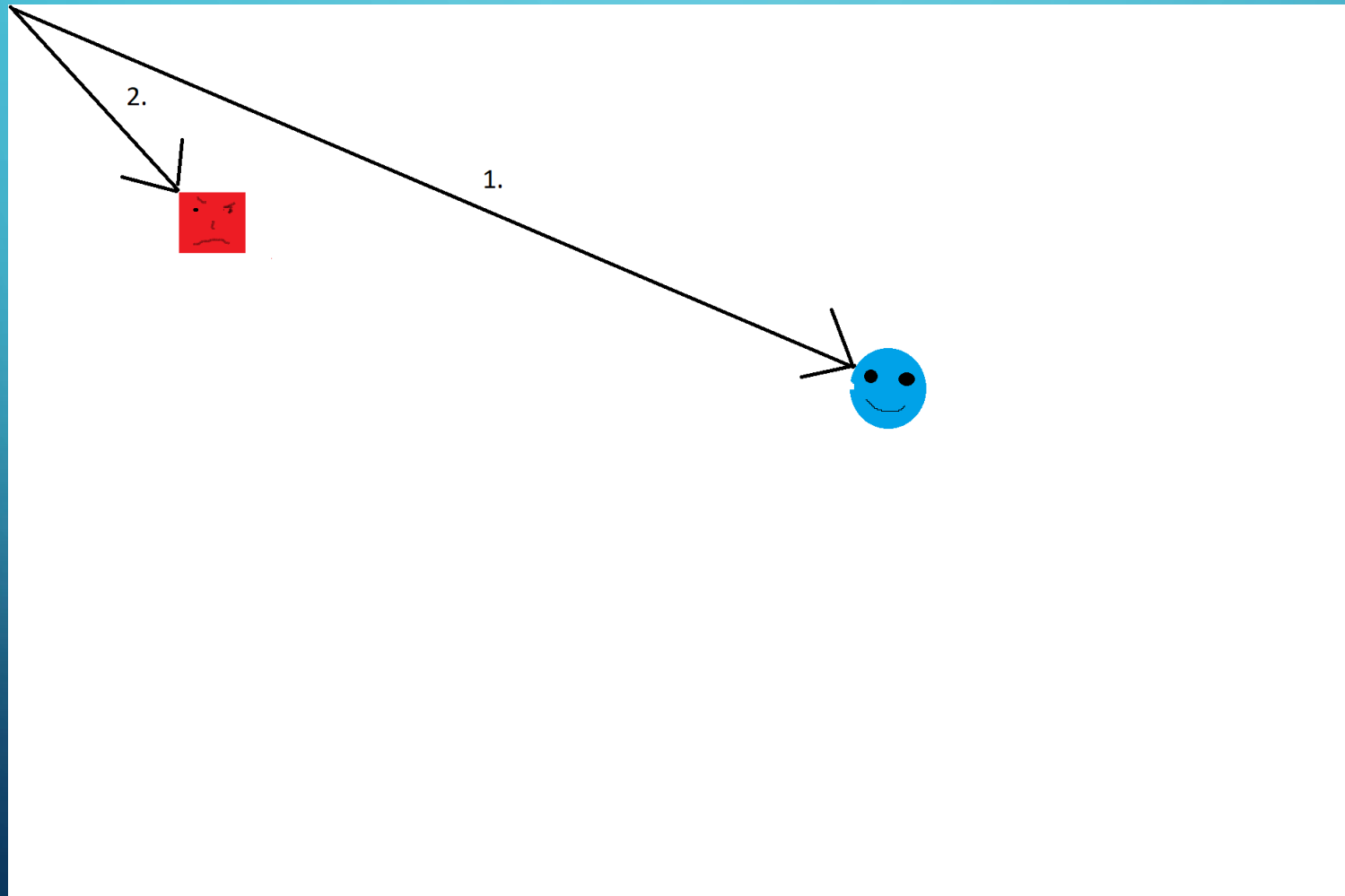
TECHNIK, PROBLEME UND LÖSUNGEN (HIER ERKLÄREN WIR NUR COOLE PROBLEME UND WIE WIR SIE GELÖST HABEN)

- Vektorgeometrie beim Programmieren
 - Steuerung des Spielers durch die Controllerklasse
- 
- 

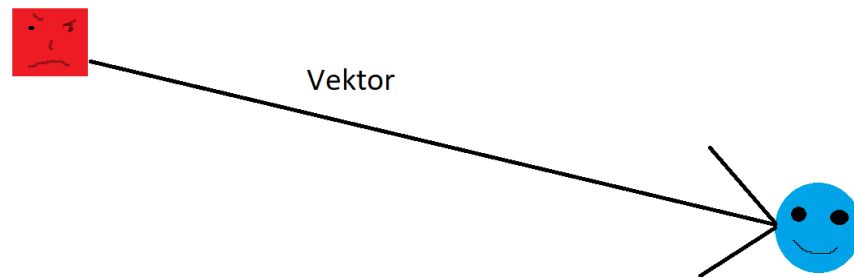
VEKTORGEOMETRIE BEIM PROGRAMMIEREN



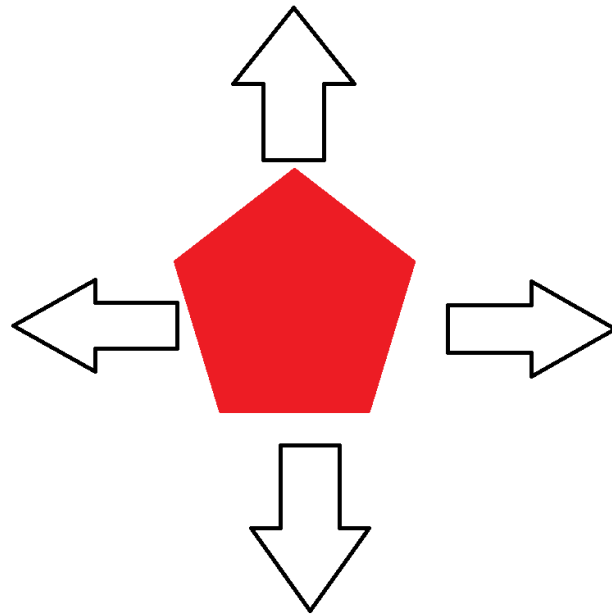
VEKTORGEOMETRIE BEIM PROGRAMMIEREN



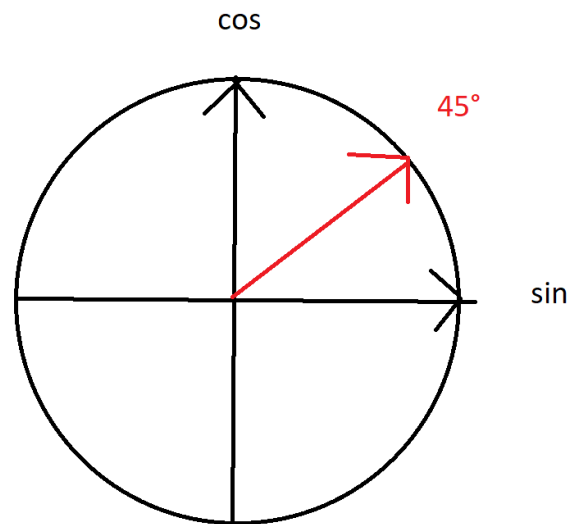
VEKTORGEOMETRIE BEIM PROGRAMMIEREN



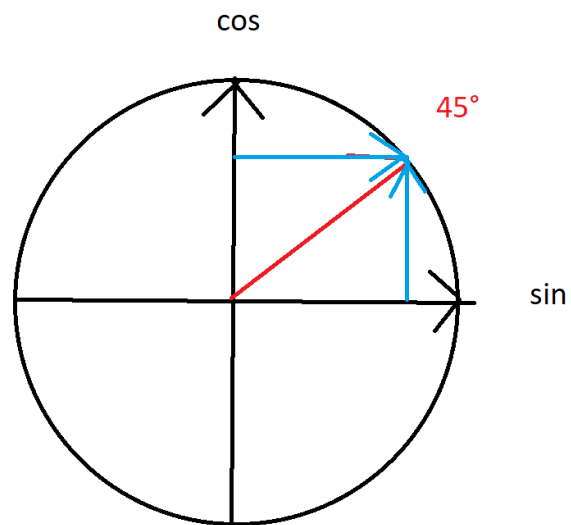
VEKTORGEOMETRIE BEIM PROGRAMMIEREN



VEKTORGEOMETRIE BEIM PROGRAMMIEREN



VEKTORGEOMETRIE BEIM PROGRAMMIEREN



Vektor($\cos(x)$, $\sin(x)$)



STEUERUNG DES SPIELERS DURCH DIE CONTROLLERKLASSE

WURDEN ALLE ZIELE ERREICHT?

- Es ist ein Multiplayer-Spiel
- Es ist 2D top-down
- Es ist (relativ) übersichtlich
- Jeder kann es schnell lernen
- Es ist ohne großen Aufwand spielbar
- Macht es Spaß?

DEMO

- Ob es Spaß macht dürfen sie selbst entscheiden

QUELLEN

- <https://godotengine.org/>
- <https://docs.godotengine.org/en/stable/>
- https://www.youtube.com/watch?v=WRDI2gQObg8&list=PL1td_Fr5vMGO_W0hasVEYlvfdm_oYh0xi9
- https://www.youtube.com/watch?v=WRDI2gQObg8&list=PL1td_Fr5vMGO_W0hasVEYlvfdm_oYh0xi9
- <https://godotforums.org/>