Vektor Klasse

|  |
| --- |
| namespace zauberbild { |
|  |  |
|  | export class Vector { |
|  | public x: number; |
|  | public y: number; |
|  | public draggable: true; |
|  |  |
|  |  |
|  | constructor(\_x: number, \_y: number) { |
|  | this.x = \_x; |
|  | this.y = \_y; |
|  | } |
|  |  |
|  | public static getRandom(\_minLength: number, \_maxLength: number): Vector { |
|  | let vector: Vector = new Vector(0, 0); |
|  | let length: number = \_minLength + Math.random() \* (\_maxLength - \_minLength); |
|  | let direction: number = Math.random() \* 2 \* Math.PI; |
|  |  |
|  | vector.set(Math.cos(direction), Math.sin(direction)); |
|  | vector.scale(length); |
|  | return vector; |
|  |  |
|  | } |
|  |  |
|  | public get length(): number { |
|  | return Math.hypot(this.x, this.y); |
|  | } |
|  |  |
|  | public set(\_x: number, \_y: number): void { |
|  | this.x = \_x; |
|  | this.y = \_y; |
|  | } |
|  |  |
|  | public add(\_addend: Vector): void { |
|  |  |
|  | this.x += \_addend.x; |
|  | this.y += \_addend.y; |
|  | } |
|  |  |
|  | public scale(\_scale: number): void { |
|  | this.x \*= \_scale; |
|  | this.y \*= \_scale; |
|  | } |
|  |  |
|  |  |
|  |  |
|  | public copy(): Vector { |
|  | return new Vector(this.x, this.y); |
|  | } |
|  | } |
|  |  |
|  |  |
|  |  |
|  | } |

Klasse Form

|  |
| --- |
| namespace zauberbild { |
|  |  |
|  | export abstract class Form { |
|  |  |
|  | public position: Vector; |
|  | public rotation: number; |
|  | public radius: number; |
|  | public velocity: Vector; |
|  | public color: string; |
|  |  |
|  |  |
|  | constructor(\_position?: Vector) { |
|  |  |
|  | this.color = ""; |
|  |  |
|  | if (\_position) |
|  | this.position = \_position.copy(); |
|  | else |
|  | this.velocity = new Vector(0, 0); |
|  |  |
|  | this.radius = 25; |
|  | this.velocity = new Vector(0, 0); |
|  | this.rotation = 0; |
|  | } |
|  |  |
|  | public abstract draw(\_crc: CanvasRenderingContext2D): void; |
|  |  |
|  | public move(\_timeslice: number): void { |
|  |  |
|  | let offset: Vector = this.velocity.copy(); |
|  | //offset.scale(\_timeslice); |
|  | offset.x \*= \_timeslice \* 0.5; |
|  | offset.y \*= \_timeslice; |
|  | this.position.add(offset); |
|  |  |
|  | if (this.position.x < 0) |
|  | this.position.x += (crcMain.canvas.width); |
|  | if (this.position.y < 0) |
|  | this.position.y += crcMain.canvas.height; |
|  | if (this.position.x > (crcMain.canvas.width)) |
|  | this.position.x -= (crcMain.canvas.width); |
|  | if (this.position.y > crcMain.canvas.height) |
|  | this.position.y -= crcMain.canvas.height; |
|  | } |
|  | } |
|  | } |

Datei Server

|  |
| --- |
| import \* as Http from "http"; |
|  | import \* as Url from "url"; |
|  | import \* as Mongo from "mongodb"; |
|  |  |
|  | export namespace zauberbild { |
|  | let orders: Mongo.Collection; |
|  | let allPictures: string[] = []; |
|  | let port: number | string | undefined = process.env.PORT; |
|  | if (port == undefined) |
|  | port = 5001; |
|  | let mongoClient: Mongo.MongoClient; |
|  | let databaseUrl: string = "mongodb+srv://Luziagu:EIA2@eia2-lozyt.mongodb.net/EIA2?retryWrites=true&w=majority"; |
|  | startServer(port); |
|  | connectToDatabase(databaseUrl); |
|  |  |
|  | function startServer(\_port: number | string): void { |
|  | let server: Http.Server = Http.createServer(); // Für Server wird Port erstellt |
|  | console.log(server); |
|  | console.log("Server starting on port:" + \_port); |
|  | server.listen(\_port); //Server hört auf Port und der Port wird geöffnet |
|  | server.addListener("request", handleRequest); // Ein Event Request wird auf den Server gesetzt, der dann die Funktion HandleRequest aufruft |
|  | } |
|  |  |
|  | async function connectToDatabase(\_url: string): Promise<void> { |
|  | let options: Mongo.MongoClientOptions = { useNewUrlParser: true, useUnifiedTopology: true }; |
|  | mongoClient = new Mongo.MongoClient(\_url, options); |
|  | await mongoClient.connect(); //MongoDB soll verbunden werden |
|  | orders = mongoClient.db("Zauberbild").collection("magicPicture"); //Daten die in Ordern gespeichert wurden werden in der collection abgelegt. |
|  | console.log("Database connection", orders != undefined); |
|  | } |
|  |  |
|  | //let anyOrder: string[] = []; |
|  | async function handleRequest(\_request: Http.IncomingMessage, \_response: Http.ServerResponse): Promise<void> { |
|  | console.log("what's up?"); |
|  | console.log(\_request.url); //Wie mit der Request umgegangen wird |
|  | \_response.setHeader("content-type", "text/html; charset=utf-8"); |
|  | \_response.setHeader("Access-Control-Allow-Origin", "\*"); |
|  | if (\_request.url) { |
|  | let url: Url.UrlWithParsedQuery = Url.parse(\_request.url, true); |
|  | let spliturl: string[] = \_request.url.split("&"); |
|  | if (spliturl[0] == "/?safeImage") { |
|  | orders = mongoClient.db("Zauberbild").collection("magicPicture"); //Daten der collection zuordnen |
|  | await (orders).insertOne(url.query); |
|  | \_response.write("Picture saved"); |
|  | allPictures = []; |
|  | } |
|  | if (spliturl[0] == "/?getImage") {//ausgewählter Titel mit Titel in Datenbank abgleichen und die richtigen |
|  | //Bilddaten anfordern, raussuchen |
|  | let picture: Mongo.Cursor<any> = orders.find({name: spliturl[1]}); |
|  | await picture.forEach(showOrders); |
|  | let jsonString: String = JSON.stringify(allPictures); |
|  | jsonString.toString(); |
|  | \_response.write(jsonString); |
|  | allPictures = []; |
|  | } |
|  | if (spliturl[0] == "/?getTitles") {//alle Titel aus Datenbank raussuchen |
|  | let names: Mongo.Cursor<any> = orders.find({}, { projection: { \_id: 0, name: true }}); |
|  | await names.forEach(showOrders); |
|  | let jsonString: String = JSON.stringify(allPictures); |
|  | jsonString.toString(); |
|  | \_response.write(jsonString); |
|  | \_response.write(names.toString()); |
|  | allPictures = []; |
|  | console.log(names); |
|  | } |
|  | } |
|  | \_response.end(); //Antwort wird verschickt |
|  | } |
|  | function showOrders(\_item: Object): void { |
|  | let jsonString: string = JSON.stringify(\_item); |
|  | allPictures.push(jsonString); |
|  | } |
|  | } |