Übung 9 PrgT: Collectables 2

Aufbauend auf der letzten Übung, bauen wir den letzten Teil des Pong-Spieles.

Dieses Mal wollen wir das Spiel um aufsammelbare Objekte ergänzen. In dieser Übung, ergänzen wir die fehlenden Teile der Übung 8

1. Event mit Zeitablauf

Da die Effekte, der eingesammelten Collectables, nur für eine gewisse Zeit anhalten sollen, brauchen wir Events, die sich wieder beenden.

Erstellen sie hierfür ein CollectableEvent für Paddle 1 und 2, registrieren Sie diese entsprechend im Observer in der main.cpp und ergänzen sie die catchEvent der Paddle.cpp um die zusätzlichen cases.

Was sie im Eventfall tun, ist Ihnen überlassen (z.b. Größe oder Farbe ändern).

Um den Effekt wieder rückgängig zu machen, starten Sie in der Eventabhandlung einen *std::thread*, der eine Lambdafunktion übernimmt. Mittels *std::this_thread::sleep_for()* können sie den Thread für eine Weile warten lassen. Danach setzten sie den geänderten Wert wieder zurück. Damit der Thread den Progammablauf nicht blockiert, müssen Sie diesen noch um ein *.detach()* ergänzen.

2. CollectableManager

Um die Collectables nun zu organisieren, brauchen wir eine Klasse CollectableManager. Diese hat folgende Eigenschaften:

- ist abgeleitet von IGameObject
- einen Standartkonstruktor
- eine public Funktion *std::vector<EventType> checkCollisions(const Ball& b);*
- entsprechende update und draw Funktionen
- eine private Variable vom Typ *std::vector<std::shared_ptr<Collectable>>*
- eine Private Variable vom Typ *std::chrono::steady_clock::time_point*

Im Konstruktor weisen wir dem Zeitpunkt einen zufälligen Wert in Sekunden zu (siehe letzte Übung) und addieren diesen mit dem jetzigen Zeitpunkt (*std::chrono::steady_clock::now()*)

In *CheckCollision* prüfen wir für alle Collectables im vector, ob diese mit dem Ball kollidieren. Wenn das der Fall ist, geben wir je nach Ballrichtung das entsprechende Event zurück und setzten das Collectable auf invalid.

In der *draw()* Funktion werden alle Collectables gezeichnet.

In der *update* Funktion, prüfen wir Zwei Sachen:

1. Wenn der Zeitpunkt t später als der jetzige Zeitpunkt ist, genieren wir an einer Zufälligen stelle im Spiel ein neues Collectable und fügen es im Vector ein. Zusätzlich müssen wir einen neuen späteren Zeitpunkt generieren
2. Alle Invaliden Collectables aus dem Vector löschen. Hierfür benutzen wir die Funktionen *std::stable_partition* sowie die swap Funktion des Vektors.