**Notification Manager**

Über die Klasse *NotificationManager* können Push-Mitteilungen in das Notification Center des Geräts eingetragen werden und daraus gelöscht werden. Vom Notification Center aus werden sie zur angegebenen Zeit automatisch angezeigt.

**Push-Mitteilung erstellen**

Die Methode *createPushNotification* erstellt eine neue Push-Mitteilung im Notification Center. Als Parameter werden die Zeit, zu der die Mitteilung erscheinen soll sowie der Inhalt der Nachricht übergeben. Falls die Zeit in der Vergangenheit liegt oder die App keine Berechtigung für Push-Mitteilungen hat, wird die Mitteilung nicht erstellt. Als ID wird die Zeit verwendet, die eindeutig ist, da nie zwei Mitteilungen mit gleicher Zeit erstellt werden können.

**Push-Mitteilung löschen**

Mitteilung werden mit Hilfe der Methode *removePushNotification* wieder gelöscht. Dabei werden sie anhand ihrer ID identifiziert. Außerdem können mit der Methode *cleanPushNotifications* alle Einträger der App in das Notification Center gelöscht werden, was bei einem Neustart des Spiels passiert.

**Queue**

Die Nachrichten werden immer nur dann verschickt, wenn die App gerade nicht aktiv ist. Dabei sind wir auf das Problem gestoßen, dass in dieser Zeit keine neuen Einträge im Notification Center erstellt werden können. Um das zu umgehen, werden die Wünsche immer „auf Vorrat“ erstellt, während die App aktiv ist. Es werden für jedes Bedürfnis zwei Einträge erstellt: einer für den Start und einer für den Ablauf.

**Bedürfnisse**

|  |  |
| --- | --- |
| **ID** | **Name** |
| 0 | Hunger |
| 1 | Durst |
| 2 | Alkohol |
| 3 | Vitamine |

Der Pirat hat vier Bedürfnisse: Er will Fleisch essen, Wasser trinken, Rum trinken und seinen Vitaminbedarf mit Obst decken, um nicht an Skorbut zu erkranken.

**Initialisierung**

Die Bedürfnisse werden am Spielstart einmalig zufällig mit der Methode *initDesires* initialisiert, indem eine Warteschlange mit fünf Einträgen erstellt wird (einer pro Leben). Dabei wird jedem Eintrag eine zufällige ID (siehe Tabelle), eine Startzeit und eine Ablaufzeit zugewiesen. Für die Zeiten werden in der TypeDef Intervalle für die Zeit zwischen Start und Ablauf eines Wunsches und zwischen dem Ablauf eines Wunsches und dem Start des nächsten angegeben. In diesen Intervallen werden gleichverteilte Zufallszahlen erstellt, die auf die aktuelle Zeit bzw. die zuletzt generierte Zeit addiert werden und so die nächste Start- oder Ablaufzeit bilden. Die Ablaufzeit eines Bedürfnisses liegt also immer vor der Startzeit des nachfolgenden, sodass keine zwei Bedürfnisse gleichzeitig aktiv sein können.

**Game Loop**

Während des Spiels wird durch einen Timer in der *AppDelegate* durchgängig mit Hilfe der Methode *checkStatus* überprüft, ob gerade ein Wunsch abläuft. Dazu wird die Datenbank ausgelesen und die Ablaufzeit jedes der eingetragenen Wünsche mit der aktuellen Zeit verglichen. Falls mindestens eine Ablaufzeit erreicht ist, ändert sich der Text der Sprechblase des Piraten entsprechend und ihm wird ein Leben abgezogen. Bei nicht aktiver App wird eine Push-Mitteilung angezeigt, die den Verlust eines Lebens mitteilt. Ein neuer Wunsch wird hierbei nicht erstellt, da der Pirat jetzt ein Leben weniger hat und man deshalb auch einen Wunsch weniger scheitern lassen darf, bis der Pirat stirbt.

Im *MainGameViewController* wird geprüft, ob eine Startzeit erreicht ist. Falls das der Fall ist, erscheint eine Denkblase über dem Piraten, in der er dem Nutzer seinen Wunsch mitteilt. Falls die App nicht aktiv ist, wird eine Push-Mitteilung angezeigt, die schon vorher erstellt wurde.

**Wünsche erfüllen**

Die Bedürfnisse werden über die entsprechenden Buttons im Menü erfüllt. Sobald einer der Buttons gedrückt wird, wird durch die Methode *fulfillDesire* geprüft, ob das Item noch im Lager vorhanden ist. Wenn das der Fall ist, wird der Bestand um eins verringert - auch, wenn das Item gerade gar nicht gefordert ist.

Da die Push-Mitteilung für den Verlust eines Lebens schon erstellt ist (siehe Notification Manager), wird diese hier wieder gelöscht, wenn der Wunsch tatsächlich erfüllt wurde. Anschließend wird ein neuer zufälliger Wunsch in die Liste aufgenommen, der wie bei der Initialisierung erstellt wird, allerdings ausgehend von der Ablaufzeit des am spätesten ablaufenden Wunschs.

Für jeden erfüllten Wunsch erhält der Pirat Erfahrungspunkte. Dadurch kann er Level aufsteigen, was sein Aussehen verändert und dem Nutzer kleine Vorteile bringt.

**Tests**

Es gibt zwei Tests für die Wünsche: man kann einen Wunsch aktivieren und einen aktiven Wunsch ablaufen lassen. Neue Wünsche zu erstellen ist nicht möglich, da dies im Spiel immer nur dann passiert, wenn davor ein anderer erfüllt wurde (Ausnahme: 1. Spielstart). Wenn man einen Wunsch erstellt ohne vorher zu löschen, kann es mehr Wünsche als Leben geben, sodass der Pirat nach seinem Tot bei inaktiver App weiterhin Wünsche angibt. Außerdem kann ein neu erstellter Wunsch, der sofort aktiv ist, die Regel verletzen, dass immer nur genau ein Wunsch aktiv ist, was zu Fehlern führt.

Beim Aktivieren eines Wunsches wird der zeitlich nächste Wunsch aus der Datenbank herausgesucht. Wenn er bereits aktiv ist, passiert nichts. Wenn er allerdings noch inaktiv ist, wird die Ablaufzeit und die ID des Wunsches gespeichert und er wird komplett gelöscht. Danach wird ein neuer Wunsch mit gleicher ID und Ablaufzeit erstellt. Als Startzeit wird die aktuelle Zeit plus 3 Sekunden festgelegt. Der Grund dafür, dass der Wunsch gelöscht und neu erstellt wird ist, dass das Verändern eines Wunsches, besonders mit den zugehörigen Push-Mitteilungen, deutlich komplizierter ist.

Dadurch, dass der Wunsch erst in 3 Sekunden aktiv wird, kann man so auch die Push-Mitteilungen testen, wenn man die App direkt schließt.

Wenn man den Ablauf eines Wunsches testet, wird zuerst geprüft, ob aktuell ein Wunsch aktiv ist. Wenn nicht, wird dies im Log vermerkt und es passiert nichts. Andernfalls wird das aktuelle Bedürfnis mit der Methode *getActiveDesire* aus der Datenbank ausgelesen und zwischengespeichert. Anschließend wird- wie bei dem anderen Test- der aktive Wunsch gelöscht und ein neuer mit gleicher Startzeit und ID erstellt. Als Ablaufzeit wird die aktuelle Zeit plus 5 Sekunden festgelegt, um das Bedürfnis noch erfüllen zu können. Die Startzeit liegt dabei in der Vergangenheit, was nur bedeutet, dass keine Push-Mitteilung mehr für den Start ausgegeben wird.

Aktuelle Bedürfnisse

In der Tabelle *aktuelleBeduerfnisse* wird die Warteschlange der Wünsche des Piraten gespeichert. Diese wird wie im Kapitel Bedürfnisse beschrieben initialisiert und während des Spiels aktualisiert, was immer direkt in die Datenbank geschrieben wird. Wie beim Pirat und dem Lager werden die aktuell gespeicherten Daten mit Hilfe der *loadData* Funktion gelesen. Die Bedürfnisse werden nicht zwischengespeichert, sodass während des Game Loops immer wieder aus der Datenbank gelesen wird.