Giovanni Baratta - 0000853633 - Laboratorio 2

Per il secondo laboratorio è stato realizzato un semplice gioco 2D il cui scopo è raccogliere tutte le gocce che cadono in modo casuale. Per raccogliere le gocce si utilizza un contenitore che è possibile spostare tramite le frecce direzionali, sinistra e destra. Se si raccolgono tutte le 4 gocce si vince, altrimenti si perde.

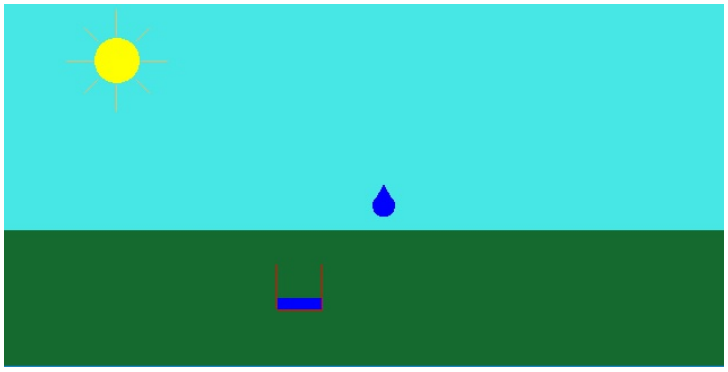

Figura 1: Schermata di gioco

Gerarchia oggetti In modo simile all'esercitazione fornita, a partire da `BaseObject` è presente una gerarchia di oggetti, ognugno dei quali è responsabile del proprio rendering. È stato definito un oggetto particolare (`Movable`), che è possibile estendere, per gli oggetti che possono modificare la propria posizione nel tempo. È poi possibile aggiungere agli oggetti delle proprietà come il colore attraverso la classe `Colorable`, ed una bouding box per le collisioni con la classe `BoundingBox`.

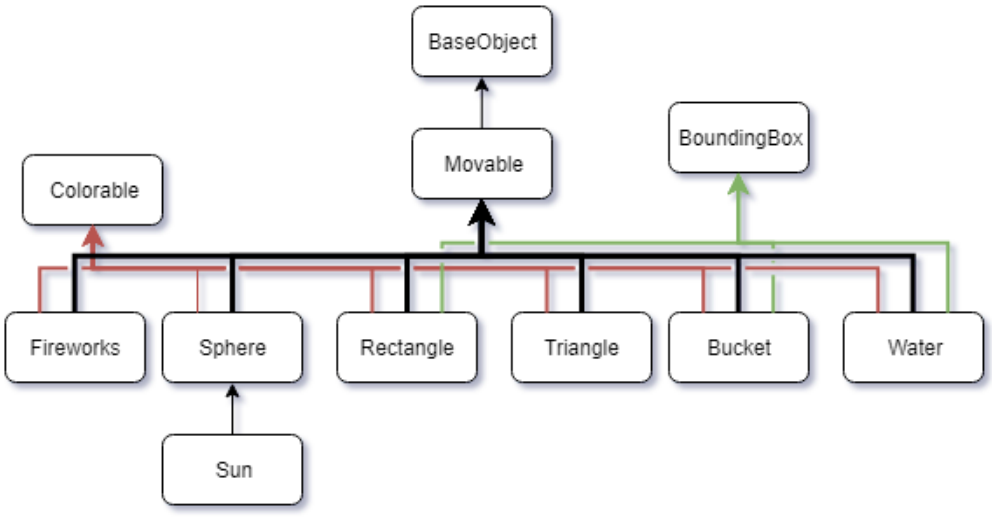

Figura 2: Gerarchia

Game manager È il componente principale che propaga gli eventi di aggiornamento di stato e di rendering agli oggetti registrato presso di lui. Si occupa di calcolare le collisioni tra gli oggetti presenti nella scena, e di verificare che la condizione di vittoria sia raggiunta. Il `main` invoca ogni 16ms i metodi `displayUpdate()` e `worldUpdate()` che effettuano rispettivamente il rendering degli oggetti in scena, l'aggiornamento dello stato di gioco. In particolare all'interno di `worldUpdate()` vengono spostati gli oggetti se necessario, vengono verificate le collisioni a seguito dei movimenti e se necessario vengono eliminati ed aggiornati gli oggetti.

Sistema particellare Al termine del gioco, se sono state raccolte tutte le gocce, viene creato un sistema particellare che simula i fuochi d'artificio. Viene creato un oggetto `Fireworks` che una volta raggiunta la posizione target, genera altri oggetti `Fireworks`. Gli oggetti fireworks sono visibili finchè non raggiungono la posizione target. Gli oggetti generati diminuisce ad ogni suddivisione.

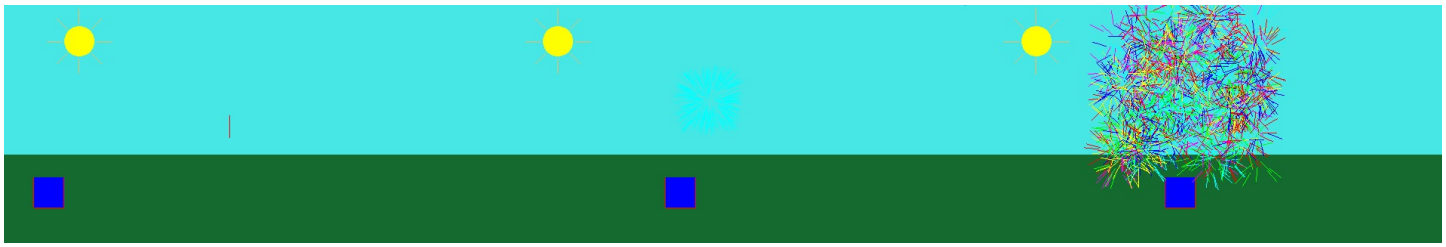

Figura 3: Sistema particellare

Movimento Per ottenere un movimento più fluido degli oggetti è stato creato un componente `SmoothTransition` che gestisce lo spostamento dell'oggetto associato ad ogni aggiornamento del mondo di gioco. Al componente viene assegnato un oggetto da gestire, una posizione target da raggiungere e un tempo per raggiungerlo. Ad ogni update, viene calcolato un vettore di spostamento e viene aggiornata la posizione del componente.

Main Oltre a registrare una funzione di callback per `glutDisplayFunc`, vengono chiamate in modo indipendete le funzioni `updateDisplay` e `updateWorld` ogni 16 ms tramite `glutTimerFunc`. Se il tempo tra una chiamata e l'altra è superiore ai 16ms, le funzioni precedenti vengono chiamate immediatamente senza aspettare ulteriormente. Le funzioni precedenti propagano l'evento al `GameManager`.