**Présentation Générale**

Il s'agit de réaliser un programme qui fait la simulation physique 2D de différents types de blocs et qui permet à l’utilisateur d’en créer et supprimer à la volée. Le programme devra respecter un certain nombre de contraintes dont le temps de latence de 20ms, le multithreading et le calcul de la physique côté serveur.

Pour son api graphique et sa simplicité de maintenance, nous avons opté pour le langage Javafx pour notre programmation.

**Objets**

Nous avons dans notre programme trois types d’objets soient :

* Cercle
* Carré
* Rectangle

Chaque objet possède une extension d’objet que l’on va appeler body qui nous permet d’y exercer des forces physiques comme la gravité, les frictions, les attractions, les propulsions, etc...

Un click primaire sur un objet change sa couleur et un click secondaire le supprime.

**Physiques**

La physique dans notre projet est déclinée en plusieurs sous composants à savoir :

* Le monde
* Les bordures
* Les forces

Parler du monde revient à parler de tout ce qui concerne la gravité. Grâce à ce composant, nous pouvons créer un monde dans lequel les lois physiques pourront être appliquées, soient : la gravité, les forces, etc0…

Les bordures délimitent la zone de notre monde. On a donc en tout 4 bordures (haut, bas, gauche, droit).

Les forces sont l’ensemble de toutes le forces physiques applicables dans notre univers.

**Partie Serveur**

La partie serveur étant la partie importante de notre projet, nous y avons consacrer assez de temps.

Dans un premier temps, la réception des données. Nous avons opté pour une identification par table. La vérification du type de traitement à faire se base donc sur une commande en « String » reçue. Par exemple, la commande « CreateCircle » voudra dire, dire créer un cercle à une position donnée que l’on enverra ensuite.

Ensuite viens le traitement des données. Avant tout, les données sont stockées dans des listes permettant un accès concurrent. Ainsi, l’on peut ajouter et modifier les éléments de ces listes sans problèmes de concurrence.

Nous avons ensuite fait usage d’une dérivée du protocole UDP que nous avons implémentée afin de garder une connexion permanente au serveur et surtout pour empêcher le server d’être bloquant.

Parlant de connexion bloquante, le javafx a un grand problème avec la gestion des commande javafx depuis une boucle infinie. Nous avons donc été confronté à un problème de ce côté mais nous avons pu assez vite le régler avec un Platform.runLater() qui demande en gros au JVM d’inclure dans le rafraîchissement de l’interface notre commande.

Concernant la gestion du multi, il fallait simplement créer un thread pour chaque utilisateur connecté. Cela a également pu facilement se faire.

Au démarrage donc, l’utilisateur choisi un port de connexion et la vitesse de rafraîchissement désirée (20ms par défaut). Le serveur va ensuite lancer un listeiner sur ce port et chaque 20ms par défaut calculé la physique du monde, puis l’envoyer aux clients.

**Partie client**

Au démarrage, le client a une interface lui demandant de saisir son nom de connexion, l’adresse du serveur et le port de connexion.

Il est ensuite envoyé vers une interface de simulation.

En cliquant sur la zone de simulation, il crée un objet.

Il peut aussi modifier la gravité globale du monde sur les deux axes (X et Y).

Une déconnexion est également possible en temps voulue.

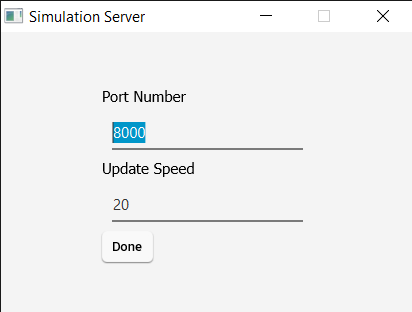
Concernant le code interne, le client fonctionne aussi par commande de table.

Un commande « Update » de la part du serveur lui dirait par exemple de refaire la mise à jour des objets, etc…

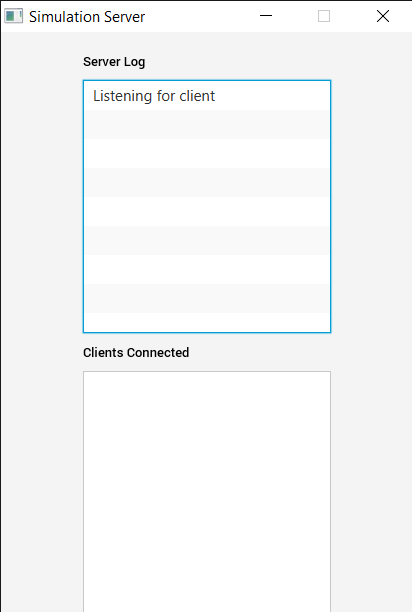
Le client a par contre une complexité. C’est qu’il ne doit pas fonctionner de la même façon que le serveur. Il peut décider quitter la partie et la rejoindre quand il veut. Mais il doit toujours avoir un rendu de l’état actuel du monde.

Là est le problème que nous avons. C’est qu’a sa connexion, vu que le server met déjà jour le monde et que son thread à lui n’est pas pris en compte, il y a un problème de mis à jour chez le nouveau client. Nous ignorons toujours la provenance de ce bug et travaillons ardemment afin de le corriger.

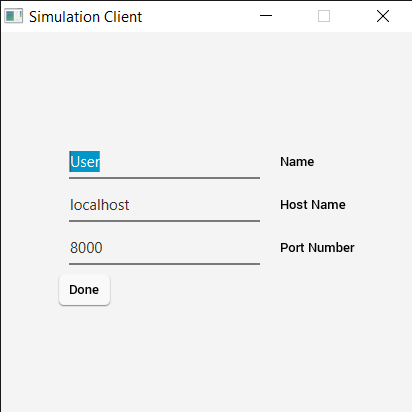
A part ce problème, nous avons aussi un problème avec la suppression des objets qui se fait un peu tardivement parfois ou qui ne se fait carrément pas.



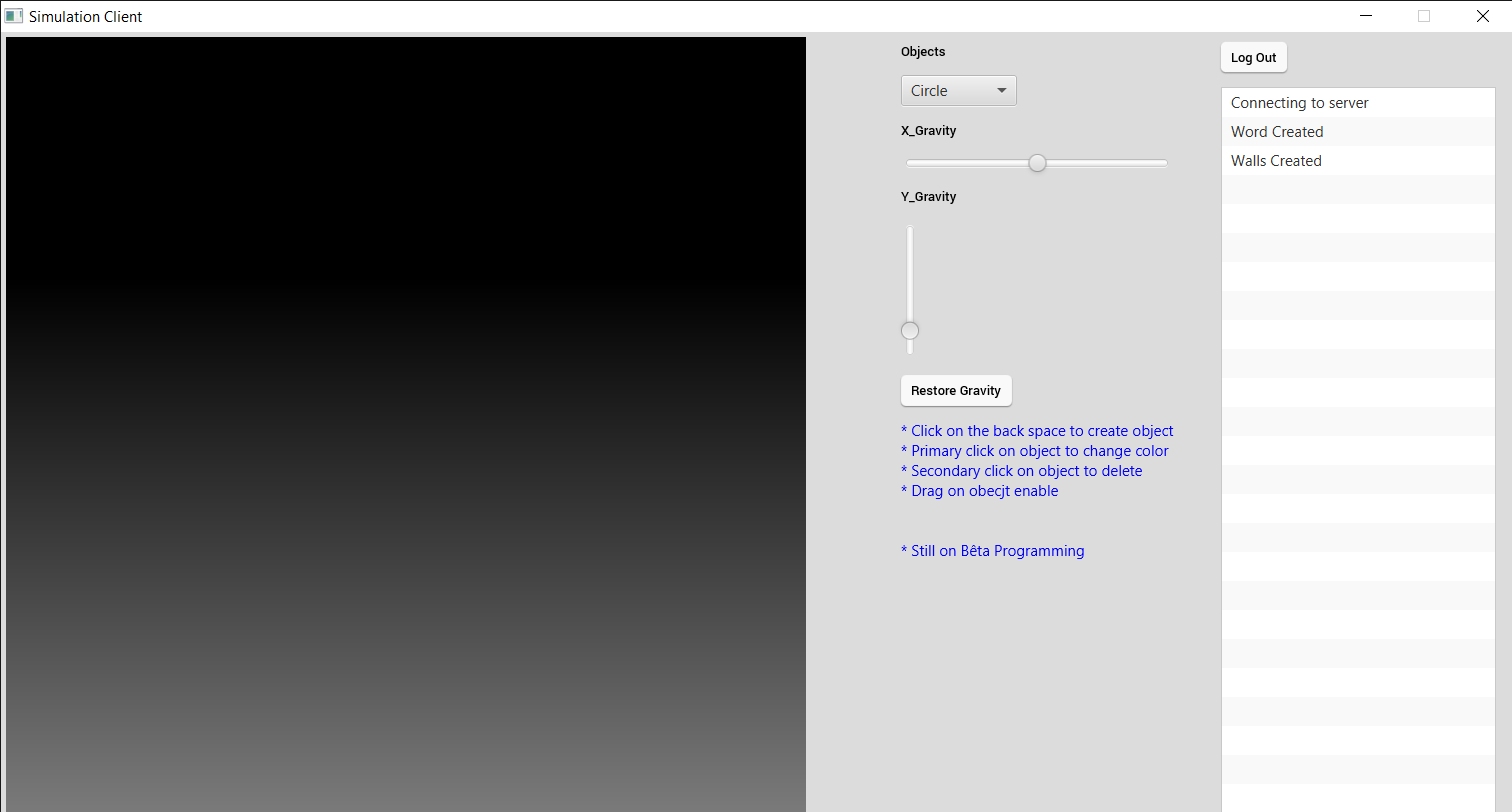
Server Launching

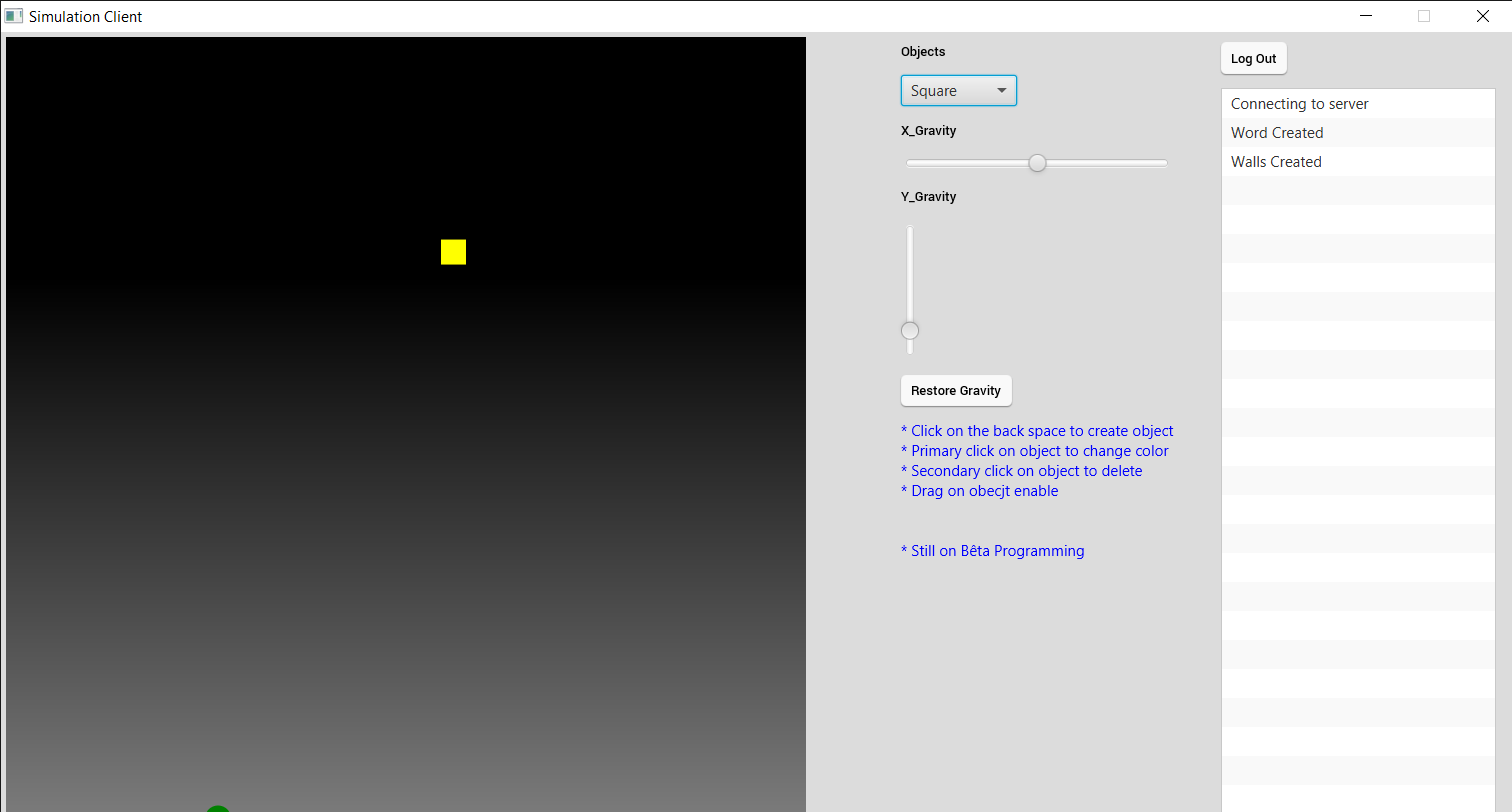
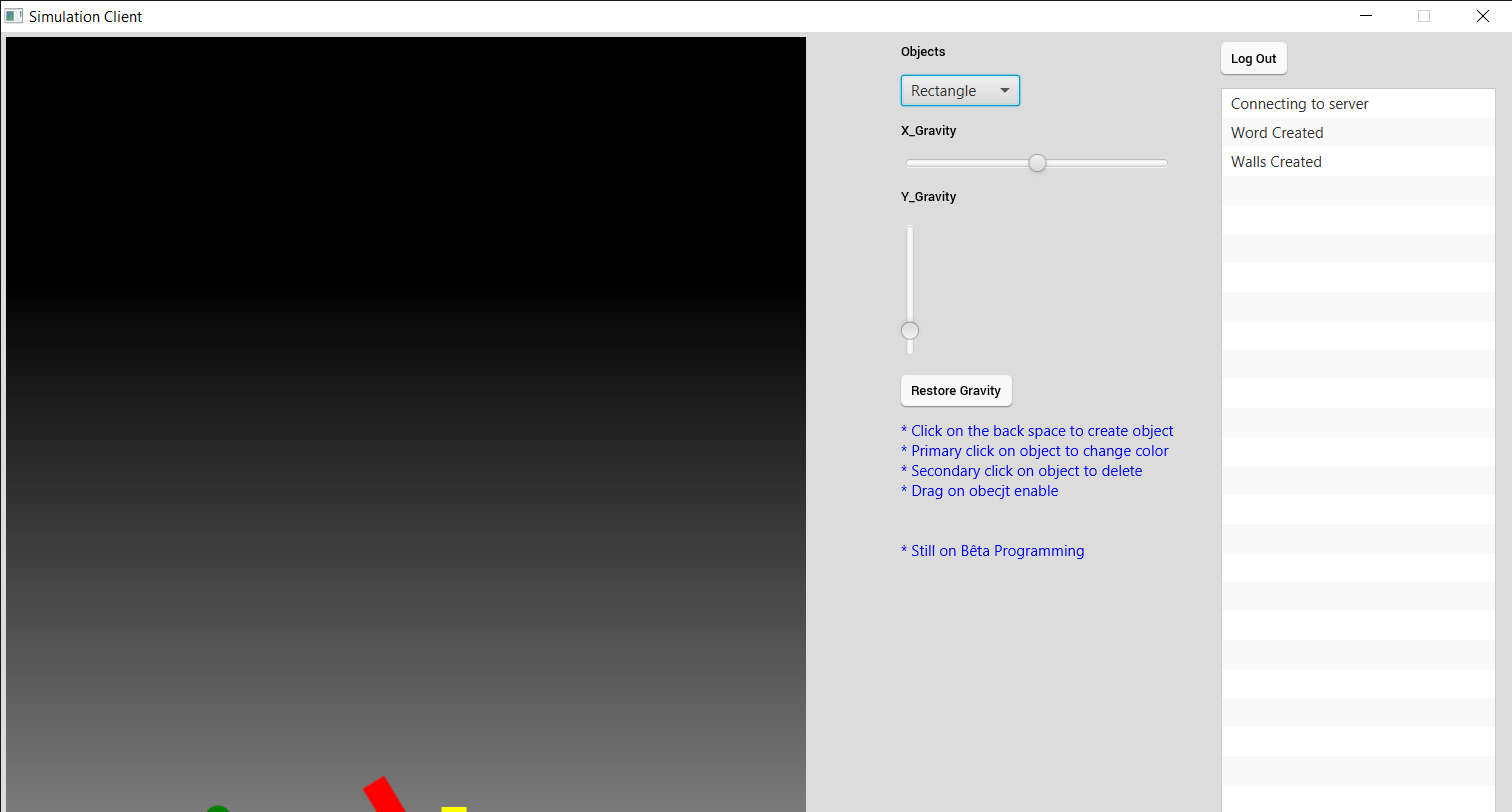
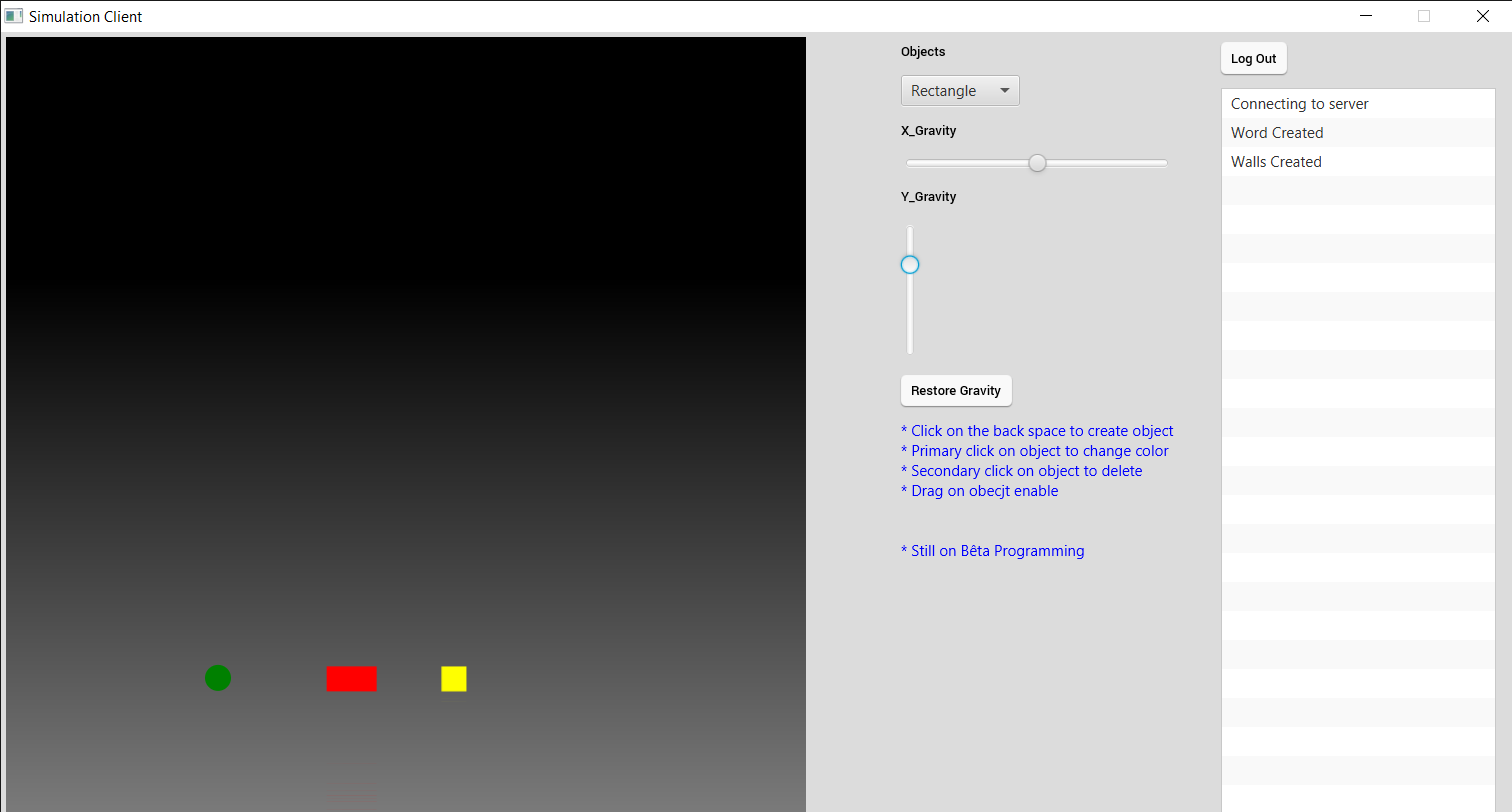


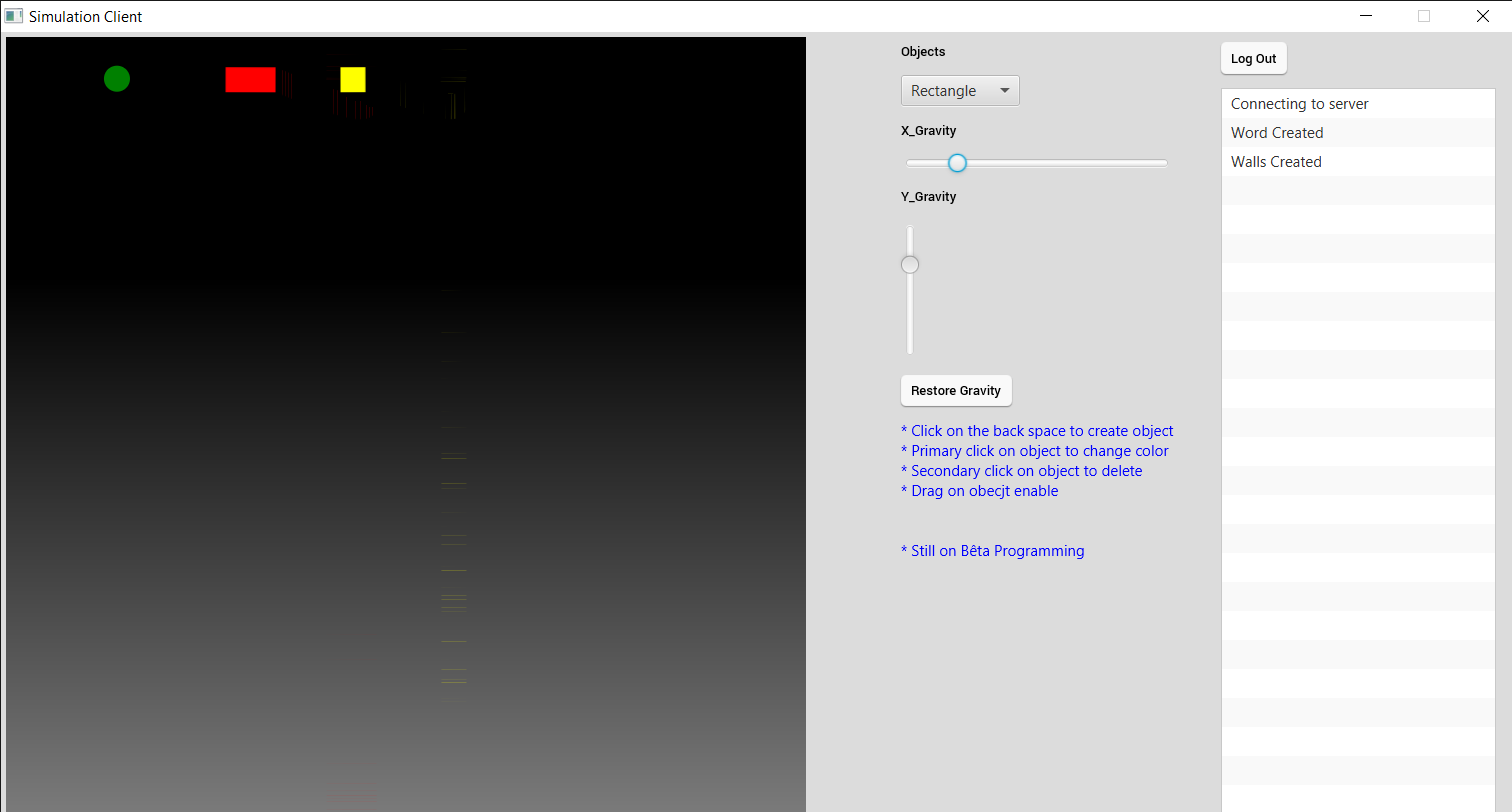
Server started



Client Starting

Client Started

Create SquareCreate Rectangle

Update Gravity Y

Update Gravity X