Chin Ho

Màrius Mora

# Pràctica de colisions – Post Mortem

L’objectiu inicial del projecte era implementar un sistema de colisions basats en l’àlgebra vectorial que estem estudiant a classe de teoría. L’objectiu bàsic era aconseguir que un cercle reboti amb un pla inclinat, tot i que a partir d’aquest punt teníem certa llibertat per expandir la pràctica i fer-la més atractiva si volíem pujar nota. Des del principi vam veure que seria fàcil transformar una pràctica en un mini-joc, de manera que vam plantejar el projecte des d’aquest punt de vista.

Hem utilitzat programació orientada a objectes ja que sabíem que seria més adequat si un cop aconseguida la funcionalitat bàsica volíem tenir-ho més fàcil per ampliar el projecte. Això ens va fer anar més lent al principi, però al final va valdre la pena. La primera decisió derivada d’aquest plantejament va ser la d’implementar les rectes com a rectangles, ja que aixi podríem crear un objecte definit on rebotin les pilotes i que no fos una sola recta infinita. Tot i que això va fer que implementar el rebot bàsic fos mot més lent ens ha permès afegir varis objectes en diferents angles de manera senzilla un cop definides les classes per als colliders dels rectangles i les pilotes.

## C:Users:mariusmora:Downloads:brainilicious:Data:brain_ball00.gif Reptes

El primer problema que vam trobar va ser fer que la pilota no entrés dins la línia, cosa que vam solucionar fàcilment fent la detecció de colisions abans del moviment i no a la inversa. La colisió amb els rectangles també ens va portar força temps ja que havies de definir les colisions am bles cantonades i que la pilota no detectés un costat del triangle com una recta infinita. Després del rebot bàsic vam implementar la gravetat, on vam trobar un bug estrany que feia que la velocitat de les pilotes anés augmentant contínuament. Vam implementar dues fórmules diferents de rebot però cap de les dues solucionava el problema, que crèiem tenía relació amb com funciona la funció draw() al processing. Efectivament, canviant l’ordre d’execució de les funcions de moviment i gravetat es soluciona el problema, tot i que la causa no acaba de quedar clara. Un altre problema relacionat amb el draw() va ser que al modificar el valor del framerate canviava la velocitat de la pilota, de manera que vam implementar un timer que controlés la velocitat a la que s’actualitzaven els moviments perquè fos independent del refresh del draw.

## C:Users:mariusmora:Downloads:brainilicious:Data:ovni_lights00.gif Encerts

El fet d’utilitzar l’orientació a objectes va fer molt fàcil incloure molts objectes en la pantalla i que un cop definides les colisions les pilotes i els rectangles es comportéssin de manera consistent. Tenim dues formules que detecten colisions amb linies o rectangles (bounceWithLine()) i colisions entre pilotes (bounceWithBall()); Hem inclòs moviment d’un rectangle controlat pel jugador (dreta, esquerra i saltar), cosa que fa la demo molt més entretinguda per l’usuari. Si les pilotes colisionen amb el rectangle del jugador desapareixen i s’incrementa un contador de puntuació, i hi ha un contador de temps. Gràcies a que el processing facilita la implementació de gràfics hem utilitzat animacions per sprites i efectes de so per donarli un aspecte de videojoc i convertir el que sembla una demo aburrida de una colisió entre una recta i un pla en un petit minijoc. Això ens ha ajudat a entendre com el processing utilitza els inputs de l’usuari i també a utilitzar llibreries externes per implementar els sons. Quan el temps s’acava apareix una pantalla de game over amb possibilitat de replay.

# Captures de pantalla



