2D Physics engine

Fredrik Bondza, dt06fb2
Simon Lindgren, dt05sl0

9 September 2009

Innehåll

1 Description

The goal is to implement a simple 2D physics engine on a Digilent Nexys2 FPGA board. The engins should be capable of simulating basic physics. For example all objects in the existing world should be affected by gravity.

2 Physics engine

2.1 Gravity

All objects which is part of the simulated world should be affected as in the real world. This means that all objects will be accelerated towards the bottom of the screen which will represent the ground in the real world.

2.2 Collision detection

In the real world two objects can not occupy the same space at the same time nor can they pass right through eachother. In order to prevent this, collision detections is necessary to simulate in a physics simulator.

Everytime a collision between two objects occur the objects shall react to the collision. For example when a ball hits the ground it bounces back up again and the world does not move since the difference in mass is extremely large. When a shape is dropped in the world it will bounce of the ground and eventually, after a few bounces, stop and stay on the ground. This means that with every collision the objects looses some of its kinetic energy, this must also be handled otherwise all objects will keep on moving forever.

2.3 World

The simulated world will be a closed world which means that none of the objects existsing in the world should be able to fall through the world. In real life most objects are substantially smaller and lighter than the earth. Since the mass of the world is much larger than that of any objects which will be simulated in the world a collision between the world and an object will not affect the world.

3 I/O processes

There will be two different I/O processes for the engine to handle, an input and an output.

The input will come from a keyboard. An keyboard event will generate an interrupt which will then be handled by the processor.

The output will be sent to a VGA controller. Since 60Hz is a refresh rate supported by most screens the graphical representation of all the objects in the physics engine should ideally be shown to the user 60 times per second. The VGA controller will be implemented as an hardware IP for high performance.

4 Memory

Since the physics engine will need to draw the simulation process as an output it will need to store matrices in the memory. The matrices will be of size 640x480 pixels to match the resolution of the VGA output.

The on-chip memory, BRAM, will not be large enough to store the matrices needed to match the VGA resolution.

The SDRAM on the FPGA board will be used to store information about the images instead of the BRAM since the SDRAM is larger than the BRAM. Using the SDRAM is slower than the on-chip BRAM but since the SDRAM has a read/write cycle of just 70ns it should be fast enough to store all the data needed.

5 Possible improvements

It might be possible to improve preformance by using more than one processor. For example the engine performance could benefit from being run on two processors where one could handle input and output and the other processor could handle all physics calculations.

By implementing as many of the computational intensive parts of the physics simulation as possible in hardware the of the physics engine could be increased.

To reduce the amount of calculations needed for each simulation step all objects could be divided into two separate groups. One group containing all moving objects and the other group contains all non-moving objects. Since the non-moving objects will never hit another object it is not necessary do check this object against collision with all other objects. Instead every moving object is checked for collision against all objects in the world.

It could be possible to reduce the amount of data sent from the processor to the VGA controller by just sending information about object positions instead of sending the whole matrix which the VGA controller should draw.

6 Time and task planning

The following table contains all the major parts of the project and the estimated working time needed for each task.

Task	*Time estimation*	*Responsible*
Physics engine	10 days	Fredrik Bondza
VGA / Keyboard controller	10 days	Simon Lindgren
Create GUI	7 days	Fredrik Bondza
Port physics engine to board	4 days	Simon Lindgren
Hardware acceleration	7 days	Simon Lindgren
Final report and presentation	7 days	Fredrik Bondza