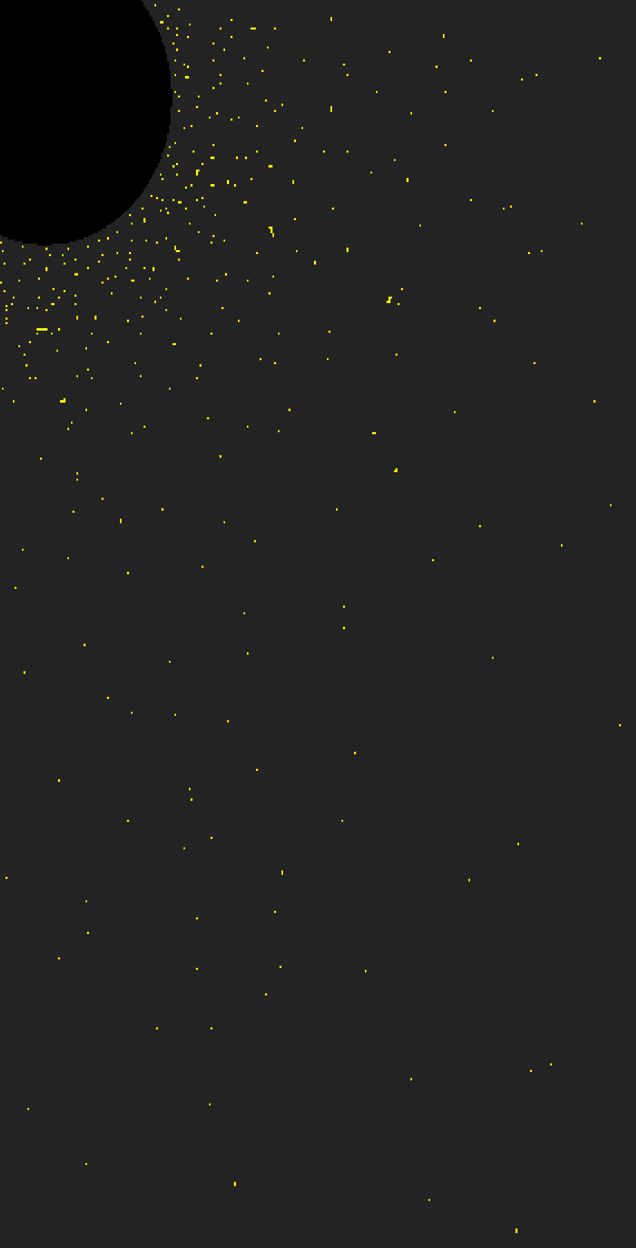


BLACK HOLE

Press ENTER to continue

Black Hole

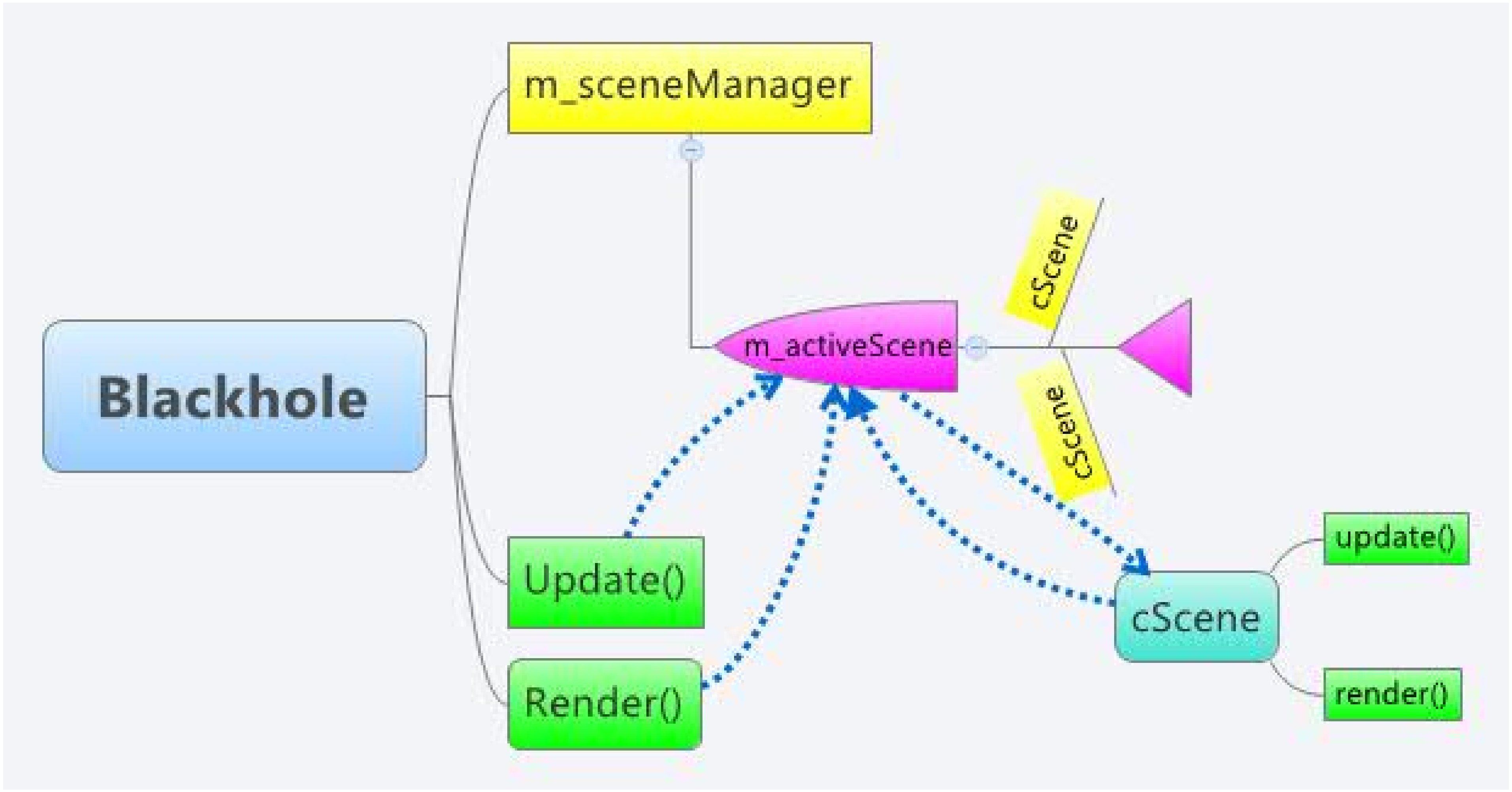
A Windows game using the Boost Libraries for threading.

The player controls a black hole using the arrow keys.

The sun in the centre of the screen emits light particles which the player must collect by colliding with them.

Collect enough particles before the sun goes supernova.

Structure of Black Hole



Laptop Specs

- AMD Phenom II P820 Triple-Core Processor 1.80GZ
- 4.00GB RAM
- 64-bit operating system
- Windows 7

Main Game Loop

- Create new light particles roughly every two frames

- Calculate velocities of all light particles
- Move light particles using new velocities
- Check all light particles for collisions with black hole

- Move black hole using keyboard input
- Check for collision between black hole and sun

- Check for end game conditions

- Render all sprites

Two update functions

UpdateLightParticles()

- Calculate velocities
- Move particles
- Detect collisions
- Update black hole mass
- Deletes particles that have left the screen

UpdateBlackhole()

- Move black hole using keyboard input
- Detect collisions with sun
- Alter black hole position if necessary

Boost 1.46.1

```
#include <boost/thread.hpp>
#include <boost/thread/barrier.hpp>
#include <boost/thread/mutex.hpp>
#include <boost/bind.hpp>
#include <boost/foreach.hpp>

boost::thread_group m_threads;
boost::barrier * m_barrier;
boost::mutex m_mass_mutex;
```

Creating a new thread

- thread_group::create_thread(Function)
- Use Boost::Bind to solve issue of using method as thread function
- Use Boost::Ref to pass references

```
m_threads.create_thread(boost::bind(&cPlayScene::UpdateLightPar
ticles, this, boost::ref(m_lightparticles[i])));
```

Using a barrier to synchronise threads

- Initialise barrier with number of threads you wish to wait for
- m_barrier->wait();
- Each thread that hits wait() decreases the barrier count
- When it reaches 0, all threads continue and the barrier is reset

Example of using barrier

Barrier(2)

Thread A()

- Calculates value X
- Barrier->wait()

Thread B()

- Barrier->wait()
- Uses value X

Using a Mutex

- Create single (global) mutex
- Inside function you wish to lock, use a scoped_lock to lock the mutex
- When function goes out of scope, mutex is automatically unlocked
- Used to protect the black hole mass when it is increased

Performance Measurements

- Timing the threaded section of the update function only
- Totalling the time over 5000 loops, using exit condition of number of new particles created to monitor this
- Storing time in global variable once game play has ended
- Will take average of 10 runs
- Currently set to have up to 5 threads by changing a global variable