Powerboat RaCe

Κάλλης Νίκος Π04050

Καραγεώργος Βαγγέλης Π04199

Πανεπιστήμιο Πειραιώς | Τμήμα Πληροφορικής | Ακαδημαϊκό Έτος 2007- 2008

Περιεχόμενα

1 Λειτουργία παιχνιδιού

1.1 Εκτέλεση

1.2 Intro scene

1.3 Gameplay

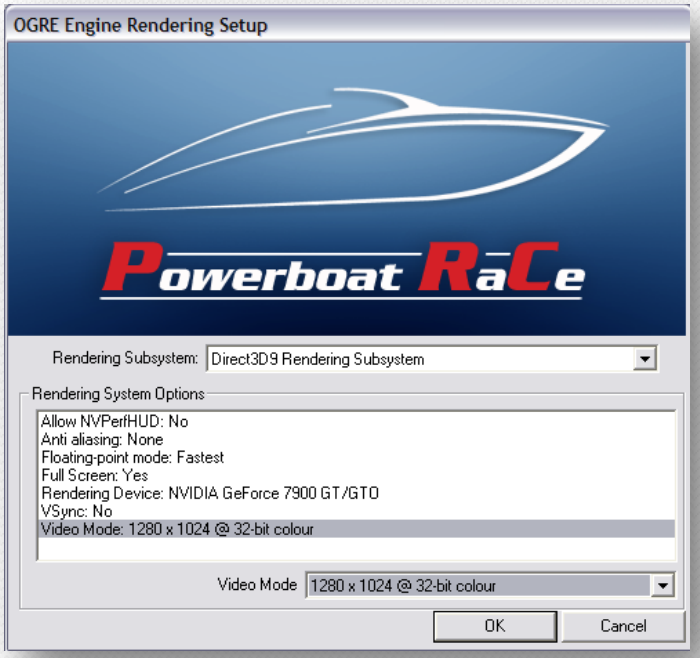
- 1.3.1 Χειρισμός
- 1.3.2 Κάμερες
- 1.3.3 Ήχος

1.4 Γραφικά

- Redistributables
- OpenAL (oalinst.exe)
- Visual Studio 2005 sp1 (vcredist_x86.exe)

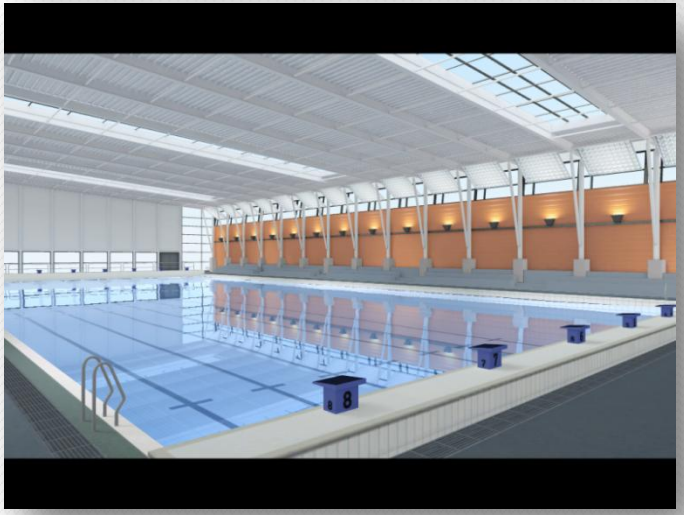
- Render system configuration
- Direct3D / OpenGL
- Windowed / Full screen
- Resolution
- Anti-aliasing
- Vsync

- ogre.cfg



- Game logo

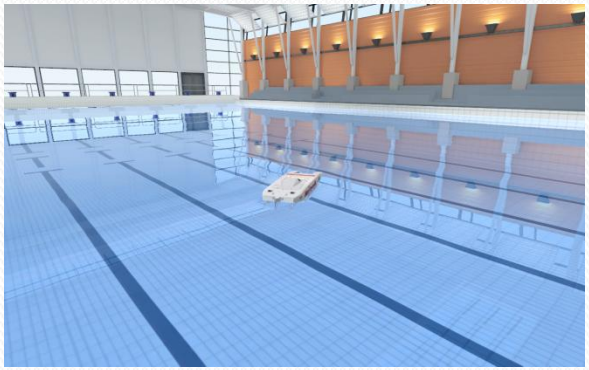
- Intro camera animation
  - Intro κάμερα
  - Keyframe animation
  - Μαύρες μπάρες (cut scene)



- Πλήκτρα χειρισμού

↑	Επιτάχυνση σκάφους	W	Κίνηση free κάμερας εμπρός
↓	Επιβράδυνση σκάφους (πορεία όπισθεν)	S	Κίνηση free κάμερας πίσω
←	Στροφή σκάφους αριστερά	A	Κίνηση free κάμερας αριστερά
→	Στροφή σκάφους δεξιά	D	Κίνηση free κάμερας δεξιά
+	Αύξηση απόστασης chase κάμερας	Q	Κίνηση free κάμερας πάνω
-	Μείωση απόστασης chase κάμερας	E	Κίνηση free κάμερας κάτω
1	Αλλαγή κάμερας σε chase	mouse	Free κάμερα Look – Around
2	Αλλαγή κάμερας σε static	PrtScn	Αποθήκευση Screenshot
3	Αλλαγή κάμερας σε free	P	Παύση παιχνιδιού
4	Αλλαγή κάμερας σε on board	M	Εναλλαγή του Polygon Mode
		LCtrl + F	Εμφάνιση / Απόκρυψη FPS

- Σύστημα καμερών



Chase κάμερα (1)

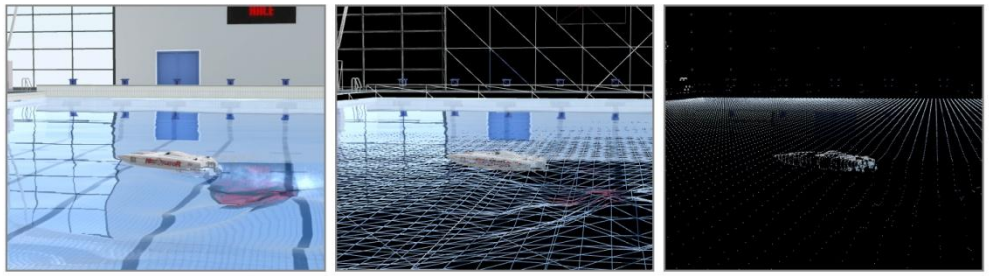
Static κάμερα (2)

Free κάμερα (3)

On-board κάμερα (4)

- Ιδιότητες καμερών
- Field of view 55˚
- On-board κάμερα FOV 80˚
- Ρύθμιση απόστασης chase κάμερας (+/-)

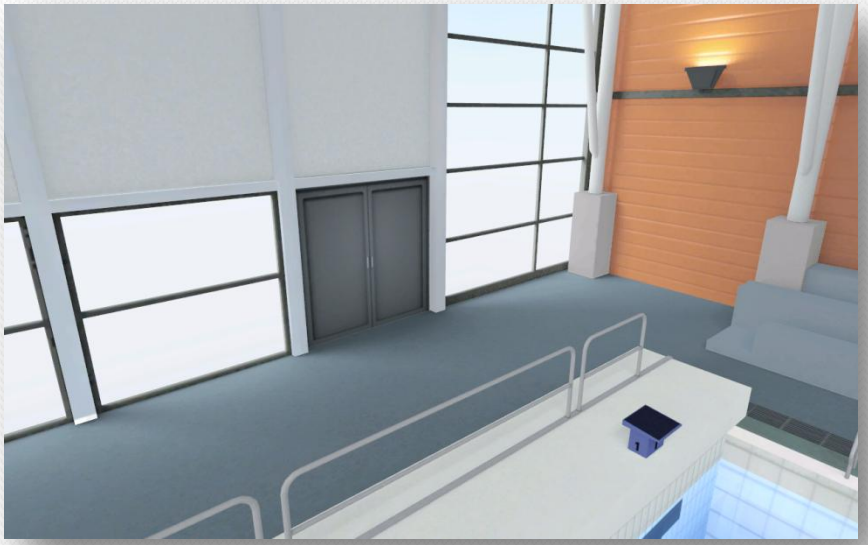
- Τρόποι προβολής πολυγώνων
- Solid
- Wireframe
- Points



- Ambient ήχος
- Χαμηλή ένταση
- Αρχείο (Ambient.wav)

- Ήχος μηχανών
- Ένταση ανάλογη με την απόσταση
- Συχνότητα ανάλογη με την ταχύτητα
- Αρχείο (Engine.wav)

- Baked textures
  - Baked lighting (global illumination)



- Νερό
  - Shaders (Fresnel.cg)
  - Reflection / refraction
  - Fresnel effect
  - Depth effect

- Dynamic cube reflections
- Shaders (EnvMap.cg)

- Skybox

2 Development software

2.1 Software & IDE’s

2.2 Engines

2.3 Physics test environment

- Modeling & textures
  - 3d studio Max 9
  - Vray renderer
  - OgreMax exporter (3ds max plug-in)
  - Nvidia DDS tools
  - Photoshop CS3

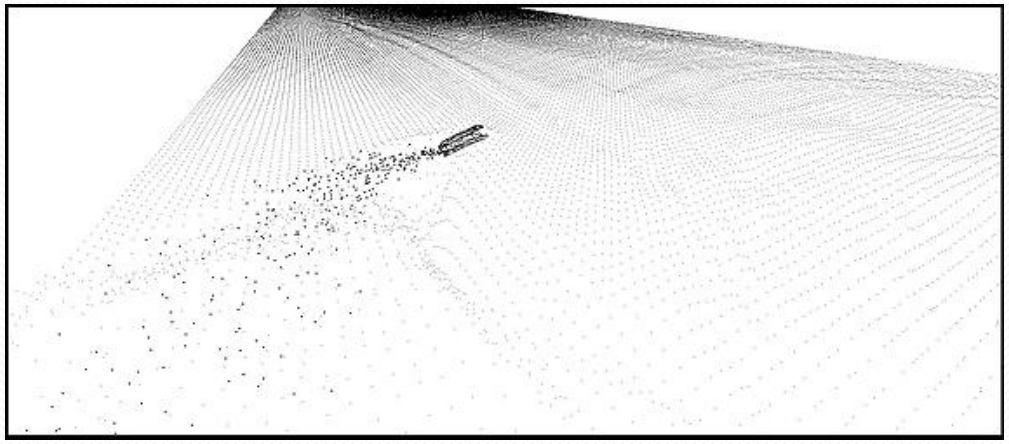
- IDE's
  - Dev C++
  - Visual Studio 2005

- Γραφικά
- Object – Oriented Graphics Rendering Engine (OGRE)

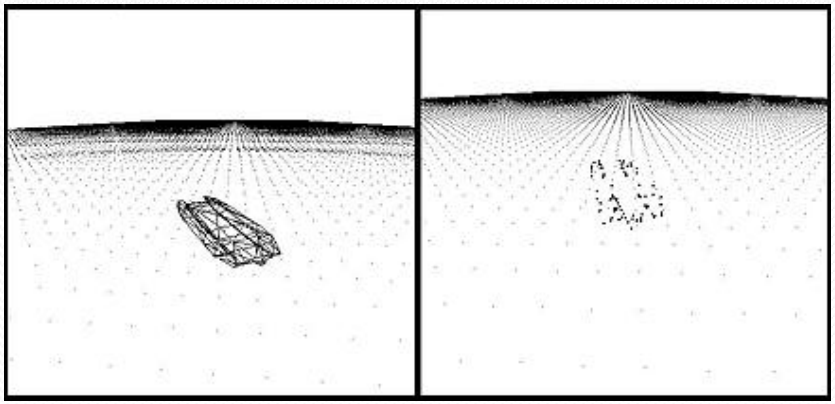
- Ήχος
- openAL
- ogreAL (wrapper)

- Input
- Object – Oriented Input System (OIS)

- Test environment
- Κίνηση νερού
- Κίνηση σκάφους
- Κίνηση chase κάμερας
- Particles



- Windows GDI



3 Artwork

3.1 References

3.2 3D Models & textures

3.3 Lighting

3.4 Materials

3.5 Overlays

3.6 Particles

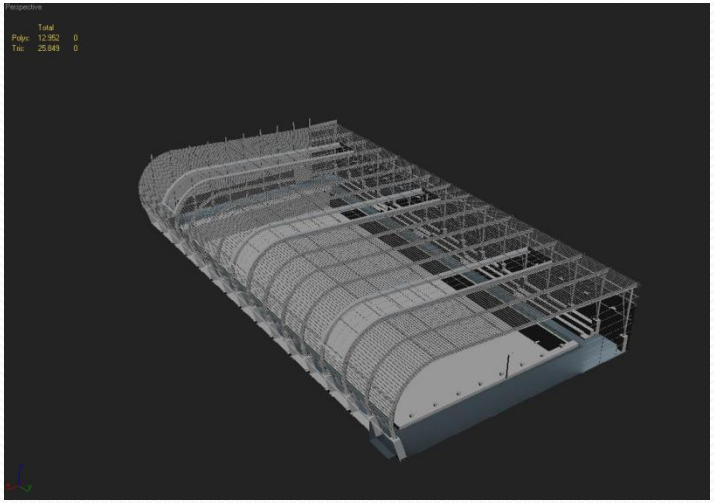
3. Artwork > 3.1 References

Σκάφος : http://www.mariosrcmodels.com/chris_christo03.htm

Πισίνα: http://sdc.lboro.ac.uk/facilities/pool/index.php?cat_id=18&subcat_id=38&level=2

- Σκάφος
- 8.365 πολύγωνα
- 2 textures (1024 χ 1024)

- Πισίνα
- 12.952 πολύγωνα
- 14 textures (2048 x 2048)



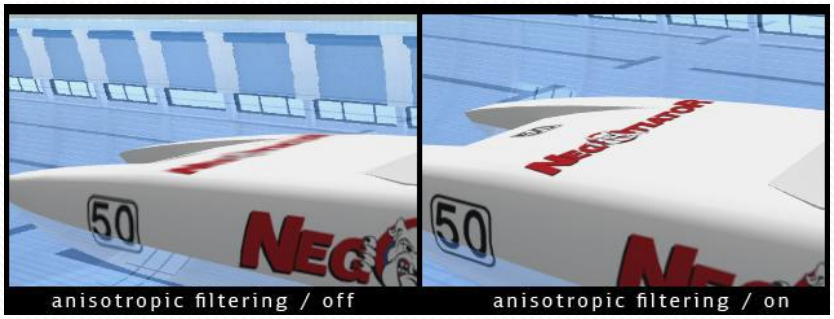
- Textures
- DDS DXT1 συμπίεση
- Light baking (vray renderer)

- Ambient light
- r: 0.9 g: 0.9 b: 0.9

- Directional light
- Πισίνα : lighting off
- Σκάφος: lighting on

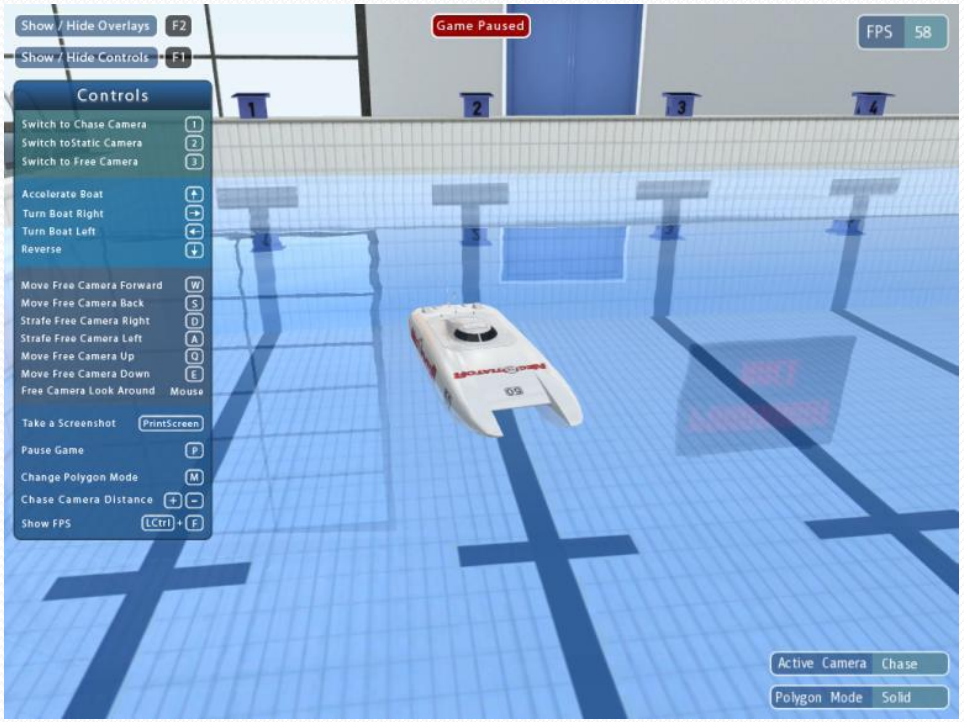
- Skybox
- Σταθερή απόσταση
- 6 textures
- Lighting off

- OGRE Materials
  - Scripts (.material)
  - Hard-coded
  - GPU Programs (Shaders)



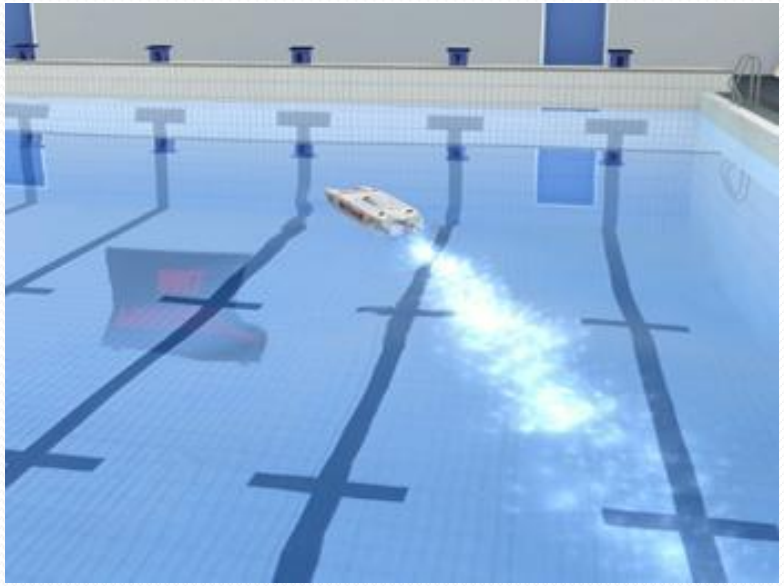
- Dynamic cube reflection mapping
  - 6 RTT's (render to texture) / frame
- Anisotropic filtering
- Material νερού
  - Ref κάμερα (clipping planes)
  - Reflection & refraction RTT /frame
  - Shader (fresnel effect, blue tint, ripples)

- Σημεία εφαρμογής
  - Game logo
  - Μαύρες μπάρες Intro scene
  - Game paused
  - Active Camera
  - Polygon Mode
  - FPS
  - Show / Hide Overlays
  - Show / Hide Controls
  - Controls



- Χαρακτηρισικά
  - Διαφάνεια (.png)
  - Μέγεθος – Ανάλυση

- Υλοποίηση
  - Billboards
  - Ημιδιαφανή
  - Γρήγορη εναλαγή
- Σημεία εφαρμογής
  - Ταραγμένο νερό
  - Καπνός εξατμήσεων
- Χρόνος ζωής
  - Εμφάνιση
  - Αύξηση μεγέθους – διαφάνειας
  - Εξαφάνιση



4 Source code

4.1 To Powerboat(.h και .cpp)

4.2 To InputListener(.h και .cpp)

4.3 To Physics Classes(.h και .cpp)

4.4 To WaterMesh(.h και .cpp)

- Application entry point

```
#if OGRE_PLATFORM == OGRE_PLATFORM_WIN32
INT WINAPI WinMain( HINSTANCE hInst, HINSTANCE, LPSTR strCmdLine, INT )
#else
int main(int argc, char **argv)
#endif
{
Powerboat game;
    try {
        game.go();
    } catch( Exception& e ) {
#if OGRE_PLATFORM == OGRE_PLATFORM_WIN32
        MessageBox( NULL, e.getFullDescription().c_str(), "An exception has occured!", MB_OK | MB_ICONERROR |
            MB_TASKMODAL);
#else
        std::cerr << "An exception has occured: " << e.getFullDescription();
#endif
    }
    return 0;
}
```

- Κλάση Powreboat
- Βασικό initialisation
- Ξεκίνημα παιχνιδιού

- Powerboat::Go()

```
void Power$boat::go()
{
		createRoot();
		soundManager = new OgreAL::SoundManager();
		defineResources();
		setupRenderSystem();
		createRenderWindow();
		initializeResourceGroups();
		createScene();
		checkPrerequisites();
		setupInputSystem();
		createFrameListener();
		startRenderLoop();
}
```

- Συναρτήσεις
- createScene() - αρχικοποιεί σκηνή, κάμερες, φυσική, ήχο και ορισμενα materials
- startRenderLoop() - είσοδος στο βασικό loop του παιχνιδιού
- Οι υπόλοιπες αρχικοποιούν την Ogre, την OgreAL, και το OIS

- Ενδεικτικός κώδικας της createScene()

- Φόρτωση σκηνής

```
//load ogremax scene
scene = new OgreMaxScene;
mWindow = mRoot->getAutoCreatedWindow();
scene->Load("scene/scene.scene", mWindow);
```

- Αρχικοποίηση φυσικής

```
//Create World
MyGrid.Create(-12.352, -26.608, 12.15, 23.86, -0.395,5);
MyBoat.ReadBof("Scene/Karina.bof",.25);
MyWorld.AddObject(MyBoat);
MyWorld.AddGrid(MyGrid);
MyWorld.Init();
MyWorld.NextState();
```

- Αρχικοποίηση ήχου

```
OgreAL::Sound *sound = soundManager->createSound("Engine", "Engine.wav", true);
sound->setRelativeToListener(false);
boatNode->attachObject(sound);
sound = soundManager->createSound("Ambient", "Ambient.wav", true);
sound->setRelativeToListener(true);
sound->setPitch(sound->getPitch()/4);
sound->setGain(0);
sound->play();
```

- Συνάρτηση InputListener() - ο constructor
- Αρχικοποιεί το αντικείμενο της κλάσης

- Συνάρτηση frameStarted()
- Υπεύθυνη για τη βασική λογική του παιχνιδιού
- Διαβάζει είσοδο απο πληκτρολόγιο και ποντικι
- Εκτελεί τη φυσική
- Μετασχηματίζει τη σκηνή και τις κάμερες
- Εκτελείται αυτόματα από την Ogre κάθε frame πριν από το rendering

- Υπόλοιπες συναρτήσεις
- Κάνουν επιπρόσθετες ενέργειες
- Καλούνται από την frameStarted()

Ενδεικτικός κώδικας της frameStarted()

Είσοδοι χρήστη

```
// Move the boat
if (mKeyboard->isKeyDown(OIS::KC_UP)) throttle=true; else throttle=false;
if (mKeyboard->isKeyDown(OIS::KC_DOWN)) back=true; else back=false;
if (mKeyboard->isKeyDown(OIS::KC_LEFT))right=true; else right=false;
if (mKeyboard->isKeyDown(OIS::KC_RIGHT)) left=true; else left=false;

// Move and Rotate the free camera
Vector3 transVector = Vector3::ZERO;
if (mKeyboard->isKeyDown(OIS::KC_W) && freeToMove == true) transVector.z -= mMove;
if (mKeyboard->isKeyDown(OIS::KC_S) && freeToMove == true) transVector.z += mMove;
if (mKeyboard->isKeyDown(OIS::KC_A) && freeToMove == true) transVector.x -= mMove;
if (mKeyboard->isKeyDown(OIS::KC_D) && freeToMove == true) transVector.x += mMove;
if (mKeyboard->isKeyDown(OIS::KC_Q) && freeToMove == true) transVector.y += mMove;
if (mKeyboard->isKeyDown(OIS::KC_E) && freeToMove == true) transVector.y -= mMove;
```

Ενδεικτικές αλλαγες φυσικής, σκηνής και ηχου

```
if(throttle)MyWorld->Command(THROTTLE);
if(back)MyWorld->Command(BACK);
if(left)MyWorld->Command(LEFT);
if(right)MyWorld->Command(RIGHT);
CreateSmoke();
CreateWater();
UpdateSmoke();
UpdateWater();
float pitch=fabs(MyWorld->Object[0]->Gas*10)+.35+boatspeed/1.5;
engSound->setPitch(pitch);
MyWorld->NextState();
Vector3 temp;
temp=bNode->getPosition()-pPos;
float v=sqrt(temp.x*temp.x+temp.y*temp.y+temp.z*temp.z);
boatspeed=boatspeed/1.5+v;
```

- Κλάση Cwavegrid
- Πλέγμα για την προσωμοίωση κυματικής μηχανικής
- Προσομοιώνει την κίνηση του νερού
- Συνδιασμός 2D κυματικής μηχανικής και συστήματος βαρών-ελατηρίων
- Συναρτήσεις αρχικοποίησης (constructor, Create() και Init())
- Array Cwavegrid::Vbuf που αντιγράφεται σε GPU hardware buffer
- Συνάρτηση της προσομοίωσης NextState()

- Ενδεικτικό κομμάτι της NextState()

```
for(int y=0;y<Ly;y++)
    {
        for(int x=0;x<Lx;x++)
        {
            if(x>0&&y>0&&x<Lx-1&&y<Ly-1)
            {
                 Mom[x][y]=Mom[x][y]/Friction+Mask[x][y]
                     +((Pos[x+1][y].Py+Pos[x-1][y].Py+Pos[x][y-1].Py+Pos[x][y+1].Py)/4-Pos[x][y].Py*1.02);
                 Tp[s][x][y].Py=Pos[x][y].Py+Mom[x][y];Mask[x][y]=0;
            }
            int numPoint = y*(Lx+0) + x ;
            float* vertex = Vbuf + 3*numPoint ;
            vertex[1]=Tp[s][x][y].Py;
            i+=3;
        }
    }
```

- Κλάση Cwsobject
- Προσωμοίωση κίνησης του σκάφους
- Περιέχει μόνο τις δομές δεδομένων
- Ο αλγόριθμος προσωμοίωσης βρίσκεται στην κλάση Cpworld

- Κλάση Ccamera
- Κίνηση της chase κάμερας
- Βάρος που συνδέεται με ελατήριο στο σκάφος
- Η προσωμοίωση γινεται με την Ccamera::NextState()

- Ενδεικτικός κώδικας της Ccamera::NextState()

```
void Ccamera::NextState()
{
    float dx,dy,dz,dst,f;
    ExtraY=Distance/6+.5;
    dx=TargetX-Cam.Cx;dy=TargetY-Cam.Cy+ExtraY;dz=TargetZ-Cam.Cz;
    dst=sqrt(dx*dx+(dy)*(dy)+dz*dz);
    if(dst>Distance)f=250;else f=500;
    if(dst>Distance||true){Chx+=dx*(dst-Distance)/f;Chy+=dy*(dst-Distance)/f;Chz+=dz*(dst-Distance)/f;}
    dst=sqrt(dx*dx+dy*dy+dz*dz);
    PrevCam=Cam;
    {Cam.Cx+=Chx;Cam.Cy+=Chy;Cam.Cz+=Chz;}
        if(Cam.Cx<BeginX)Cam.Cx=BeginX;
        if(Cam.Cx>EndX)Cam.Cx=EndX;
        if(Cam.Cy<BeginY)Cam.Cy=BeginY;
        if(Cam.Cy>EndY)Cam.Cy=EndY;
        if(Cam.Cz<BeginZ)Cam.Cz=BeginZ;
        if(Cam.Cz>EndZ)Cam.Cz=EndZ;
        if(Cam.Cy<Limy)Cam.Cy=Limy;
    Chx/=1.215;Chy/=1.215;Chz/=1.215;
    dy=TargetY-Cam.Cy;
    dx=dx/dst;dy=dy/dst;dz=dz/dst;
    Look(Cam,dx,dy,dz);
}
```

- Κλάση Cpworld
- Βασική μηχανή φυσικής
- Περιέχει αντικείμενα των Cwavegrid , Cwsobject, Ccamera
- Υπολογίζει τη φυσική του συστηματος νερό-σκάφος-κάμερα
- Βασικό interface με Cpworld::NextState() και Cpworld::Command()

- Ενδεικτικό κομμάτι κώδικα της Cpworld::NextState()

```
x3=(Object[o]->Weight[p2].Px)-(Object[o]->Weight[p1].Px);
y3=(Object[o]->Weight[p2].Py)-(Object[o]->Weight[p1].Py);
z3=(Object[o]->Weight[p2].Pz)-(Object[o]->Weight[p1].Pz);
dist=sqrt(x3*x3+y3*y3+z3*z3);
r=(dist-L);r=r;
x3=x3/dist;y3=y3/dist;z3=z3/dist;
f=r*I/Object[o]->Weight[p1].W;
if(f>.1)f=.1;
Object[o]->Weight[p1].Mx+= +x3*f;
Object[o]->Weight[p1].My+= +y3*f;
Object[o]->Weight[p1].Mz+= +z3*f;
f=r*I/Object[o]->Weight[p2].W;
if(f>.1)f=.1;
Object[o]->Weight[p2].Mx+= -x3*f;
Object[o]->Weight[p2].My+= -y3*f;
Object[o]->Weight[p2].Mz+= -z3*f;
if(Type==0)
{
    r=dist-L;
    x3=x3/dist;y3=y3/dist;z3=z3/dist;
    AddForce(Mx1,My1,Mz1,W1,-x3*r*I,-y3*r*I,-z3*r*I)
    AddForce(Mx2,My2,Mz2,W2,x3*r*I,y3*r*I,z3*r*I)
}
```

- Κλάση WaterMesh
- Δημιουργεί το mesh του νερού
- Υπολογίζει τα face & vertex normals κάθε frame
- Μεταφέρει τα δεδομένα του νερού σε Hardware buffer

- Ενδεικτικός κώδικας
- Συνάρτηση updateMesh() - Ανανεώνει τα δεδομένα του νερού ανα frame

```
void WaterMesh::updateMesh()
{
    calculateNormals();
    // set vertex buffer
    posVertexBuffer->writeData(0,
        posVertexBuffer->getSizeInBytes(), // size
        vertexBuffer, // source
        true); // discard?
}
```

- Συνάρτηση CalculateNormals() - Ανανεώνει τα normals του νερού ανα frame

```
void WaterMesh::calculateNormals()
{
    int i,x,y;
    float *buf = vertexBuffer;
    for(i=0;i<numVertices;i++)vNormals[i] = Vector3::ZERO;
    buf = vertexBuffer;
    unsigned int* vinds = (unsigned int*) indexBuffer->lock(0, indexBuffer->getSizeInBytes(),
            HardwareBuffer::HBL_READ_ONLY);
    float *pNormals = (float*) normVertexBuffer->lock(0, normVertexBuffer->getSizeInBytes(),
            HardwareBuffer::HBL_DISCARD);
    for(i=0;i<numFaces;i++) {
        int p0 = vinds[3*i] ;int p1 = vinds[3*i+1] ;int p2 = vinds[3*i+2] ;
        Vector3 v0(buf[3*p0], buf[3*p0+1], buf[3*p0+2]);
        Vector3 v1(buf[3*p1], buf[3*p1+1], buf[3*p1+2]);
        Vector3 v2(buf[3*p2], buf[3*p2+1], buf[3*p2+2]);
        Vector3 diff1 = v2 - v1; Vector3 diff2 = v0 - v1 ;
        Vector3 fn = diff1.crossProduct(diff2);
        vNormals[p0] += fn; vNormals[p1] += fn; vNormals[p2] += fn;
    }
    for(y=0;y<Pointsz;y++) {
        for(x=0;x<Pointsx;x++) {
            int numPoint = y*(Pointsx+0) + x ;
            Vector3 n = vNormals[numPoint] ;
            n.normalise() ;
            float* normal = pNormals + 3*numPoint ;
            normal[0]=n.x;
            normal[1]=n.y;
            normal[2]=n.z;
        }
    }
    indexBuffer->unlock();
    normVertexBuffer->unlock();
}
```

5 Known issues

5.1 Συμβατότητα

5.2 Ελλείψεις

5.3 Προβλήματα

- Τρέχει αποκλειστικά σε Windows

- Υψηλές απαιτήσεις
  - Υψηλός αριθμός πολυγώνων
  - RTTs / frame
  - Μέγεθος textures

- Απαιτεί σύγχρονη κάρτα γραφικών
  - Shader model 2.0

- Ηχος
- Μονοφωνικός ήχος
- Ελλειψη ειδικών εφφε (reverb, doppler)
- Δεν υπάρχει ήχος πρόσκρουσης

- Γραφικά
- Cube reflection για τα μεταλλικά αντικείμενα
- Πραγματικο Depth effect του νερού

- Φυσική
  - Το σκάφος στρίβει αργά
  - Επιπλέον σφάλματα στην κίνηση
- Γραφικά
  - Ασυνέχεια στα όρια της πισίνας
  - Μη ρεαλιστικη διάθλαση του σκάφους

The end...

Powerboat RaCe