ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Γραφικά Ηλεκτρονικών Υπολογιστών

Προγραμματιστική Άσκηση 2

Κατασκευή και οδήγηση οχήματος σε πίστα

Αρεκλάκης Άγγελος ΑΜ:809

Καραμολέγκος Χρήστος ΑΜ: 846

Περιεχόμενα

Λειτουργία προγράμματος

Οδηγίες εκτέλεσης

Για να τρέξει το πρόγραμμα αρκεί να εκτελεστεί το αρχείο «car_game.exe» που βρίσκεται στον φάκελο “Release” του Microsoft Visual Studio 2008 project. *ΣΗΜΑΝΤΙΚΟ* Απαραίτητη για την εκτέλεση του προγράμματος είναι η ύπαρξη του «glut32.dll» στον ίδιο φάκελο.

Μόλις εκτελεστεί το πρόγραμμα θα εμφανιστεί ένα ακίνητο αυτοκίνητο μέσα στην πίστα με την κάμερα κολλημένη πίσω του. Το αμάξι κινείται χρησιμοποιώντας τα βελάκια . Με το μπροστά επιταχύνει , με το πίσω επιβραδύνει , με το δεξιά στρίβει προς τα δεξιά ενώ με το αριστερά στρίβει προς τα αριστερά . Με την χρήση του κουμπιού K μπορεί να αλλάξει η κάμερα και να φαίνεται όλη η πίστα από ψηλά. Επίσης με την χρήση του Esc τερματίζεται το πρόγραμμα .

Απαιτήσεις και υλοποίηση λογισμικού

Ξεκινήσαμε τον σχεδιασμό του αμαξιού ξεκινώντας από τις ρόδες

```
for(int i = 0; i < 2; i++){
    for(int j=0; j < 2; j++) {
        glColor3f(0.0f, 0.0f, 0.0f); //
        glPushMatrix();
        glTranslatef(i*3.0,0,j * 3.0);
        glutSolidTorus(0.2,0.4,30,30);
        glColor3f(1.0f, 1.0f, 1.0f);
        gluDisk(quobj, 0, 0.2, 30, 30);
        glPopMatrix();
    }
    glPushMatrix();
    glTranslatef(i*3.0,0,0);
    gluCylinder(quobj, 0.2, 0.2, 3, 30,
    glPopMatrix();
}

glTranslatef(0 ,0.4,0.4);
for(int i = 0; i < 2; i++){
    for(int j=0; j < 2; j++) {
        glColor3f(0.9f, 0.9f, 0.9f); //
        glPushMatrix();
        glTranslatef(i*3.0,0,j * 2.2);
        glutSolidTorus(0.3,0.2,30,30);
        glPopMatrix();
    }
}
```

στην συνέχεια προσθέσαμε το κύριο μέρος του αμαξιού

```
    glPushMatrix();
    glColor3f(0, 0, 1); // blue
    glTranslatef(1.5,0.15,1.1);
    glScalef(1,0.35,0.7);
    glutSolidSphere(2.35,30,30); // body
    glPopMatrix();

    glPushMatrix();
    glColor4f(0, 0.3, 1, 0.1); // blue
    glTranslatef(1.5,0.9,1.1);
    glScalef(1,0.3,1);
    glutSolidSphere(1,30,30); // glass
    glPopMatrix();

    glPushMatrix();
    glColor3f(1, 1, 1); // white
    glTranslatef(3.6,0.1,1.1);
    glScalef(0.3,0.15,1);
    glRotatef(-30,0,0,1);
    glutSolidCube(2.35); // bumper
    glPopMatrix();

    glPushMatrix();
    glColor3f(1, 1, 1); // white
    glTranslatef(-0.7,0.4,1.1);
    glScalef(0.3,0.15,1);
    glRotatef(-30,0,0,1);
    glutSolidCube(2.35); // bumper
    glPopMatrix();
}
```

και έπειτα σχεδιάσαμε την πίστα μας δημιουργώντας 2 ορθογώνια για να μας βοηθήσει στην δημιουργία του collision box.

```
          0.0f, 1.0f,  0.0f);

    // Draw ground
glColor3f(0.3f, 0.3f, 0.3f);
glBegin(GL_QUADS);
    glVertex3f(-100.0f, 0.0f, -100.0f);
    glVertex3f(-100.0f, 0.0f,  100.0f);
    glVertex3f( 100.0f, 0.0f,  100.0f);
    glVertex3f( 100.0f, 0.0f, -100.0f);
glEnd();

glColor3f(1, 0, 0); // red
glPushMatrix();
glScalef(1.6,10,0.8);
glRotatef(90,1,0,0);
glutSolidTorus(1,50,30,12); // outer wall
glPopMatrix();

glColor3f(0.6, 0.3, 0.3); // white
glPushMatrix();
glScalef(1.7,8,0.8);
glRotatef(90,1,0,0);
glutSolidTorus(1,12,30,18); // inner wall
glPopMatrix();
```

Το επόμενο βήμα μας ήταν η επίτευξη της κίνησης του αυτοκινήτου. Υπολογίζουμε την επόμενη θέση του με βάση την γωνία σε σχέση με τον οριζόντιο άξονα, την οποία αυξομειώνει ο χρήστης με τη χρήση του αριστερού και δεξιού βέλους, και της ταχύτητας, την οποία επίσης αυξομειώνει ο χρήστης με τη χρήση του πάνω και κάτω βέλους. Η ταχύτητα παίρνει ακέραιες τιμές από 0 ως 5, με την τιμή 0 να έχει το αυτοκίνητο σε στάση και την τιμή 5 να είναι η μέγιστη ταχύτητα. Αφού υπολογίζουμε την επόμενη θέση του αυτοκινήτου με αυτόν τον τρόπο, μετακινούμαστε σε αυτή τη θέση και το σχεδιάζουμε ξανά, αφού το περιστρέψουμε κατά τη γωνία μας.

```
carx += cos(realAngle) * speed * 0.01;
carz += -sin(realAngle)  * speed * 0.01;

glTranslatef(carx,0,carz);
glRotatef(angle,0,1,0);
drawCar();
```

Στην συνέχεια αφού ορίσαμε την αρχική θέση της κάμερας

```
// XYZ position of the camera
float eyex = -5.0f, eyey = 3.0f, eyez = 25.0f;
// XYZ position of the camera center
float x2 = 0, y2 = 3, z2=20;
```

και βάλαμε να αλλάζει κάμερες με το κουμπί K.

```
{

    if (key == 'k'){
        printf("cam = %d\n",cam);
        if(cam == 1){
            eyex = 0.0f;
            eyey = 99.0f;
            eyez = 0.8f;
            x2 = 0;
            y2 = 0.0f;
            z2 = 0;
            cam = 0;
        }
        else{
            eyex = carx - 7;
            eyey = 3.0f;
            eyez = carz + 25;
            x2 = carx;
            y2 = 3.0f;
            z2 = carz + 25;
            cam = 1;
        }
    }
```

Ορίσαμε την αρχική κάμερα να ακολουθάει το αυτοκίνητο μας και να το κρατάει σταθερά στο κέντρο του πλάνου. Η θέση της κάμερας γίνεται το άθροισμα της θέσης του αυτοκινήτου με την σταθερή μεταξύ τους απόσταση μήκους 7 μονάδων.

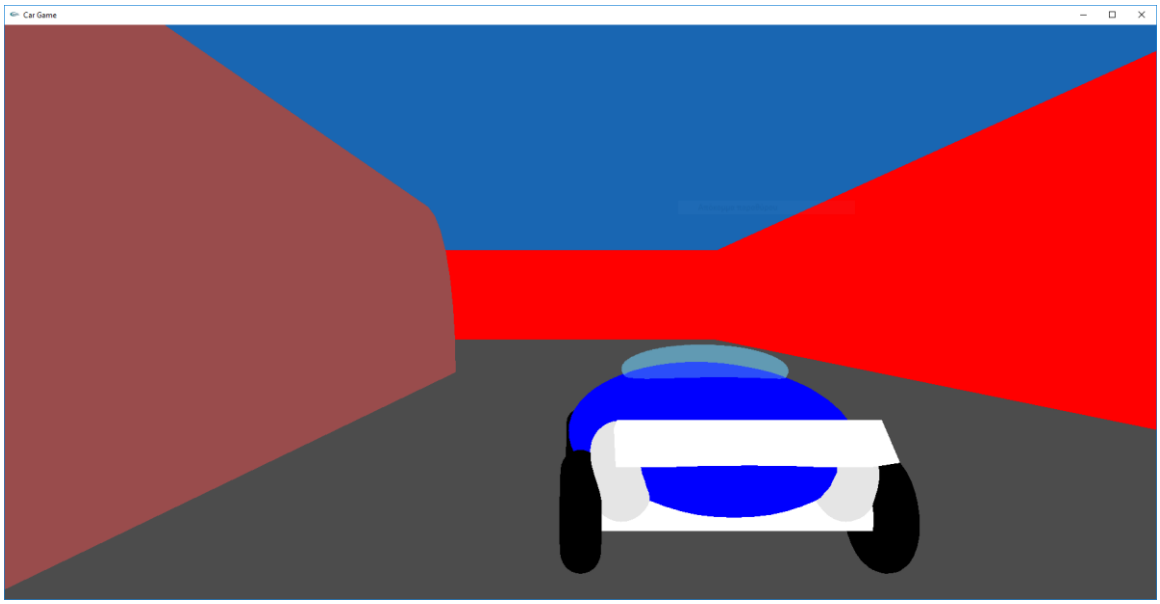
```
if(cam == 1){
    eyex = 7 * -cos(realAngle) + carx;
    eyez = 7 * sin(realAngle) + carz;
    x2 = carx;
    z2 = carz;
}
```

Τέλος, για να αποτρέψουμε την μετακίνηση του αυτοκινήτου εκτός πίστας, χρησιμοποιήσαμε την παραλληλία των τοίχων της πίστας με τον οριζόντιο και τον κάθετο άξονα. Έτσι, αφού υπολογίζουμε το σημείο της μπροστινής πλευράς του αυτοκινήτου, κοιτάμε σε ποια θέση θα βρίσκεται στην επόμενη κίνηση και αν το σημείο βρίσκεται εντός πίστας αφήνουμε το αυτοκίνητο να μετακινηθεί. Διαφορετικά, η ταχύτητα μηδενίζεται και το αυτοκίνητο περιστρέφεται κατά 90 μοίρες, ώστε να μπορεί να ξεκινήσει να κινείται και πάλι.

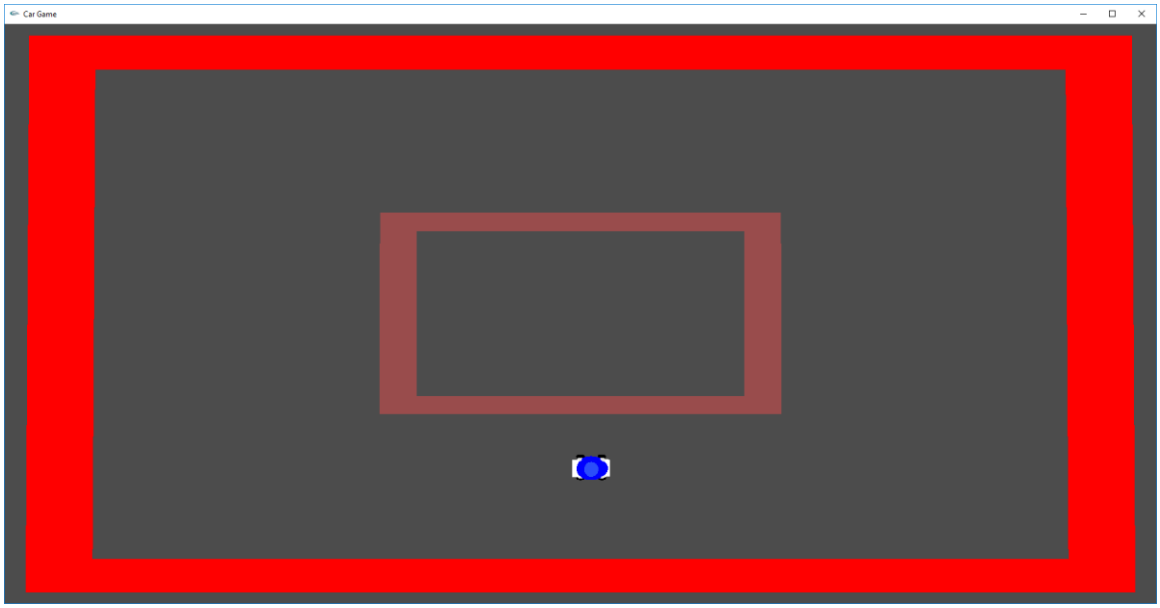
```
bool willCollide(float x , float z){

    float boxX = carx + x;
    float boxZ = carz + z;

    boxX += cos(realAngle) * 4.2;
    boxZ += -sin(realAngle)  * 4.2;

    if( ( (boxX > -28 && boxX <28 ) && (boxZ < 14 && boxZ > -14) ) || ( (boxX > 69 || boxX < -69) || (boxZ < -35 || boxZ > 35) ) )
        return true;
    return false;

}
```

```
if(willCollide(cos(realAngle) * speed * 0.01, -sin(realAngle)  * speed * 0.01)){
    speed = 0;
    angle += 90;
    if(angle >= 360) angle -= 360;
    realAngle = angle * M_PI / 180;

}
else {
    carx += cos(realAngle) * speed * 0.01;
    carz += -sin(realAngle)  * speed * 0.01;
}
```

Παραδείγματα εκτέλεσης



Αρχική θέση του αυτοκινήτου από την οπτική γωνία της πρώτης κάμερας.



Αρχική θέση του αυτοκινήτου από την οπτική γωνία της δεύτερης κάμερας.