بسم الله الرحمن الرحيم و السلام عليكم

جاري هذا الموضوع حول مجالات التحكم في عالم الأوبن جي أل , لربما كنت دائما تحاول ربط مؤشر الفأرة بعالم الأوبن جي أل الثلاثي الأبعاد , هذا الموضوع كتبته خصيصا لذلك و سأتطرق إلى كل شيء بإذن الله تعالى.

التحديد

هناك منهاج يدعى بالتحديد أي selection و هو وسيلة تمكننا من معرفة الكائن أو المجسم المتموضع في عالم الأوبن جي أل بصورة أخرى سأشرح ما يدور خلف كواليس التحديد في عالم الأوبن جي أل:
في البداية نقوم برسم المشهد و بطبيعة الحال سيكون في ذاكرة الإيطار framebuffer و بعدها نجعل التطبيق أي البرنامج يقفز من النمط الحالي التصير render و هو الإفتراضي إلى نمط آخر و هو اللتحديد selection ومن ثم نعرض المشهد , هل فهمت ما أقصد , أي أن عملية التحديد لن تحصل إلا في نمط التحديد mode selection , من المنطقي ألا يتغير محتوى ذاكرة الإيطار حيت يتم الخروج من نمط التحديد و الإنتقال إلى نمط التصيير render , عندها سيرجع السيد أوبن جي أل قائمة تحتوي على بيانات ناتجة عن تقاطع الكائنات أو المجسمات المتموضعة في عالم الأوبن جي أل مع حجم الرؤية المحدد viewing folume (حجم الرؤية هو دليل نحدده نحن وهو معرف بواسطة نمط الرؤية الحالي و مصفوفة الإسقاط و أي إضافات خاصة بمسطحات القص),كل شكل يتقاطع مع حجم الرؤية سيولد تداخل hit (عموما أعتمد في ترجمة المرادفات حسب الخاصية و ليس المعنى)و بعد ذلك يقوم السيد أوبن جي أل بتمرير المعرفات الخاصة بكل مجسم في مصفوفة من نوع integer تدعى تلك المعرفات بالأسماء names و بعض البيانات المرافقة لها .
هذه المراحل تمثل ميكانيزمة التحديد يرجى التقيد بها:
1- تخصيص مصفوفة لتحتفظ بالتداخلات و بياناتها و يتم إنشاء هذه المصفوفة عن طريق إستدعاء الدالة glSelectBuffer .
2- المرور إلى نمط التحديد عن طريق تمرير الدليل GL_SELECT إلى الدالة glRenderMode .
3- إعداد المكدس و تهيئته بأول إسم يحتل القائمة بالدالة glInitNames ثم نقوم بحجز إسم لكل شكل أو مجسم في المكدس عن طريق الدالة glPushName .
4- تعريف حجم رؤية مناسب و حجز و تحرير مصفوفات التحويل الحالية بإستعمال الدالتان glPushMatrix و glPopMatrix .
5- إعداد الأشكال و المجسمات التي ترغب في تحديدها لحظة تنفيذك للبرنامج.
6- الخروج من نمط التحديد و العودة إلى نمط التصير للقيام بمعالجة البيانات التي أرجعها السيد أوبن جي أل hit record .

```
void glSelectBuffer(GLsizei size, GLuint *buffer);
```

هذه الدالة تقوم بتخصيص مصفوفة لتقوم بالإحتفاظ ببيانات التحديد المرجعة , البارميتر buffer هو مؤشر لمصفوفة من نوع unsigned integer لتلك البيانات التي تم حفظها و البارميتر size هو عدد طبقات المصفوفة , أنت بحاجة إلى إستدعاء هذه الدالة قبل الدخول إلى نمط التحديد.

```
GLint glRenderMode(GLenum mode);
```

حقيقة هذه الدالة هي التي تجعلنا نقفز من نمط لآخر أي من نط التحديد إلى نمط التصيير و العكس (النمط الإفتراضي هو نمط التصيير render mode) , البارميتر mode يأخذ أحد التعريفات التالية:
GL_RENDER و GL_SELECT و GL_FEEDBACK (هذا الأخير يدعى بنمط التغذية الإسترجاعية هو شبيه بنمط التحديد إلا أنه يقوم بتجميد المشهد و لا يرجع سوى البيانات الخاصة بالمجسمات و ذلك لا يصب في مصلحتنا).
بالفعل سيبقى تطبيقك محافظا على النمط الذي حددته إلى غاية تغييرك له أي أن الأنماط لا تتغير تلقائيا.

بإمكانك التعرف على النمط الحالي إن إحتجت إلى ذلك عن طريق إستدعاء الدالة glGetIntegerV و تمررها التعريف GL_RENDER_MODE .

و كما لمحت من قبل فلإنشاء مكدس إسم NAME STACK نستدعي نتدعي الدالة glInitNames و هي بكل بساطة تقوم بإفراغ المكدس من كل البيانات و بعدها نضيف أسماء من نوع integer لها نسبة لكل مجسم أو شكل أو كائن مرسوم على المشهد.
الدوال التي تدير المكدس هي :
glPushName : هذه الدالة تقوم بحجز مكدس الإسم .
glPopName : هذه الدالة تقوم بتحرير مكدس الإسم .
glLoadName : إعادة تموضع الإسم في أعلى المكدس .
هذه الشيفرة تعرض ما تم شرحه

```
glInitNames();
glPushName(0);
glPushMatrix(); /* حجز التحويل الحالي */
/* هنا قد ننشئ حجم الرؤية المرغوب فيه */
glLoadName(1);
drawSomeObject();
glLoadName(2);
drawAnotherObject();
glLoadName(3);
drawYetAnotherObject();
drawJustOneMoreObject();
glPopMatrix(); /* تحرير التحويل المحجوز*/
```

في هذا المثال إن كلا الشكلين أو الكائنين الأول و الثاني يملكان إسمان مختلفان و الشكلين الثالث و الربع يشتركان في نفس الإسم , فإذا حدث تحديد لأحد الشكلان الثالث و الرابع فإن التطبيق سيرجع الإسم المحدد أو الممثل لذلك الشكل.
قد يكون هناك تعدد لأشكال تشترك كلها في إسم واحد إذا لم تكن تود التفصيل.

```
void glInitNames(void);
```

وظيفة هذه الدالة هو إفراغ و تهيئة مكدس الإسم تماما.

```
void glPushName(GLuint name);
```

تقوم هذه الدالة بحجز إسم name بداخل مكدس الإسم , و قد يحدث الخطأ GL_STACK_OVERFLOW إذا كان حجم المكدس لا يتسع لذلك , و أيظا لو أنك كنت في حاجة إلى عمق المكدس فعليك إستدعاء الدالة glGetIntegev مع تمرير المعرف GL_NAME_STACK_DEPTH و هذا للحصول على عمق المكدس الحالي.

```
void glPopName(void);
```

هذه الدالة تقوم بتحرير إسم واحد فقط من أعلى مكدس الإسم , و قد يتولد الخطأ GL_STACK_UNDERFLOW إذا ما حاول التطبيق تحرير إسم بينما المكدس فارغ.

```
void glLoadName(GLuint name);
```

هذه الدالة تعيد القيمة إلى قمة مكدس الإسم مع الإسم name , قد تحدث الدالة glLoadName الخطأ GL_INVALID_OPERATION إذا ما كان مكدس الإسم فارغا بعد إستدعاء الدالة glInitNames .

إن إستدعاء أحد الدوال الثلاثة glPushName و glPopName و glLoadName تكون غير فعالة في حالة كان التطبيق ليس في نمط التحديد .

التداخلات hit records

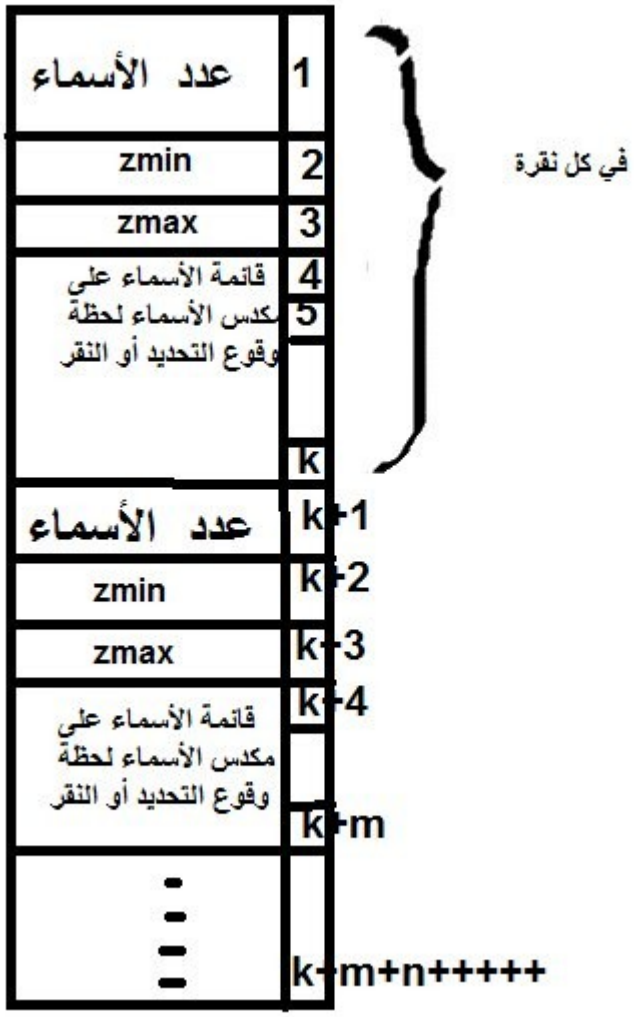
بعد خروج التطبيق من نمط التحديد و الدخول إلى نمط التصيير يقوم السيد أوبن جي أل بإرجاع عدد التداخلات و بعض البيانات في المصفوفة المحدد بالدالة glSelectBuffer من الظاهر أن لدى هذا النوع من البيانات ذاكرة خاصة به . و يتكون كل تداخل من أربع بيانات و بالترتيب التالي:

1- عدد الأسماء في مكدس الإسم أين ظهر التداخل.

2- كلا من قيمتا إحداثيات النافذة ل z القصوى zmax و الدنيا zmin لذلك التداخل .

3- قائمة لأسماء الكائنات أو الأشكال أين حدث التحديد أو النقر عليها (سيأتي موضوع النقر).

و بحيث أن التطبيق يدخل في كل مرة نمط التحديد سيقوم الأوبن جي أل بإعداد المؤشر نحو بداية المصفوفة و هذا كل ما تكتب بيانات التداخل في المصفوفة أي بمعنى آخر يتم تحديث المؤشر.



مثال عن التحديد

```
void drawTriangle(GLfloat x1, GLfloat y1, GLfloat x2,
GLfloat y2, GLfloat x3, GLfloat y3, GLfloat z)
 {
   glBegin(GL_TRIANGLES);
     glVertex3f(x1, y1, z);
     glVertex3f(x2, y2, z);
     glVertex3f(x3, y3, z);
   glEnd();
```

```
}
void drawViewVolume(GLfloat x1, GLfloat x2, GLfloat y1,
GLfloat y2, GLfloat z1, GLfloat z2)
{
  glColor3f(1.0, 1.0, 1.0);
  glBegin(GL_LINE_LOOP);
     glVertex3f(x1, y1, -z1);
     glVertex3f(x2, y1, -z1);
     glVertex3f(x2, y2, -z1);
     glVertex3f(x1, y2, -z1);
  glEnd();
  glBegin(GL_LINE_LOOP);
    glVertex3f(x1, y1, -z2);
    glVertex3f(x2, y1, -z2);
    glVertex3f(x2, y2, -z2);
    glVertex3f(x1, y2, -z2);
  glEnd();
  glBegin(GL_LINES); /* 4 lines */
    glVertex3f(x1, y1, -z1);
    glVertex3f(x1, y1, -z2);
    glVertex3f(x1, y2, -z1);
    glVertex3f(x1, y2, -z2);
    glVertex3f(x2, y1, -z1);
    glVertex3f(x2, y1, -z2);
    glVertex3f(x2, y2, -z1);
    glVertex3f(x2, y2, -z2);
  glEnd();
}
void drawScene(void)
{
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluPerspective(40.0, 4.0/3.0, 1.0, 100.0);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  gluLookAt(7.5, 7.5, 12.5, 2.5, 2.5, -5.0, 0.0, 1.0, 0.0);
  glColor3f(0.0, 1.0, 0.0); /* green triangle */
     drawTriangle(2.0, 2.0, 3.0, 2.0, 2.5, 3.0, -5.0);
  glColor3f(1.0, 0.0, 0.0); /* red triangle */
     drawTriangle(2.0, 7.0, 3.0, 7.0, 2.5, 8.0, -5.0);
  glColor3f(1.0, 1.0, 0.0); /* yellow triangles */
     drawTriangle(2.0, 2.0, 3.0, 2.0, 2.5, 3.0, 0.0);
     drawTriangle(2.0, 2.0, 3.0, 2.0, 2.5, 3.0, -10.0);
     drawViewVolume(0.0, 5.0, 0.0, 5.0, 0.0, 10.0);
}
void processHits(GLint hits, GLuint buffer[])
{
  unsigned int i, j;
  GLuint names, *ptr;
  printf("hits = %d\n", hits);
  ptr = (GLuint *) buffer;
  for (i = 0; i < hits; i++)
     { /* for each hit */
         names = *ptr;
         printf(" number of names for hit = %d\n", names); ptr++;
         printf(" z1 is %g;", (float) *ptr/0x7fffffff); ptr++;
         printf(" z2 is %g\n", (float) *ptr/0x7fffffff); ptr++;
         printf(" the name is ");
         for (j = 0; j < names; j++)
             { /* for each name */
                  printf("%d ", *ptr); ptr++;
             }
         printf("\n");
     }
}
#define BUFSIZE 512
void selectObjects(void)
{
  GLuint selectBuf[BUFSIZE];
  GLint hits;
  glSelectBuffer(BUFSIZE, selectBuf);
  (void) glRenderMode(GL_SELECT);
  glInitNames();
  glPushName(0);
    glPushMatrix();
```

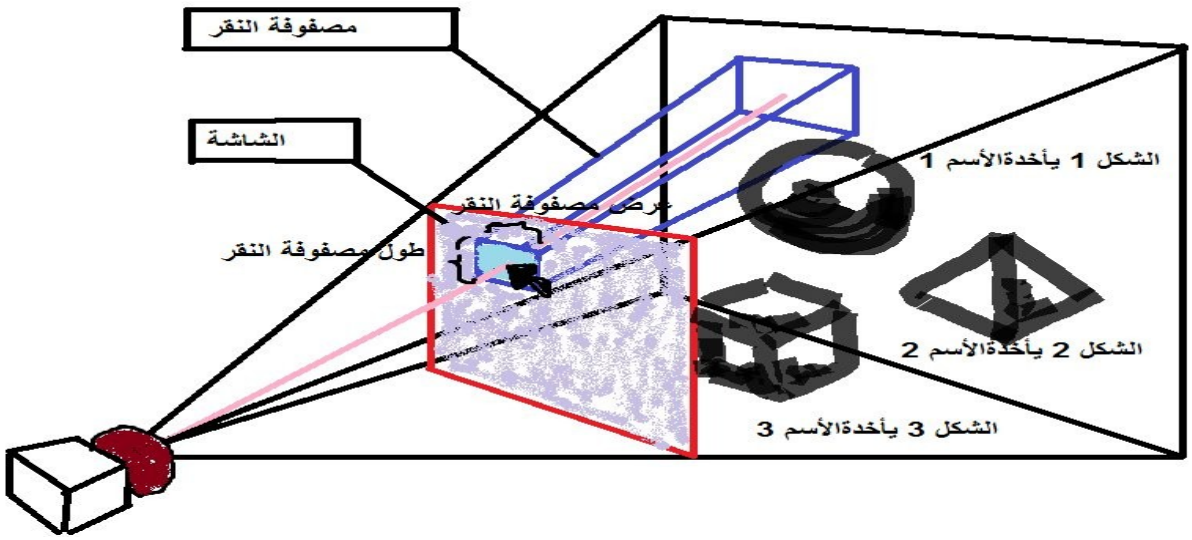
```
       glMatrixMode(GL_PROJECTION);
       glLoadIdentity();
       glOrtho(0.0, 5.0, 0.0, 5.0, 0.0, 10.0);
       glMatrixMode(GL_MODELVIEW);
       glLoadIdentity();
       glLoadName(1);
         drawTriangle(2.0, 2.0, 3.0, 2.0, 2.5, 3.0, -5.0);
       glLoadName(2);
         drawTriangle(2.0, 7.0, 3.0, 7.0, 2.5, 8.0, -5.0);
       glLoadName(3);
         drawTriangle(2.0, 2.0, 3.0, 2.0, 2.5, 3.0, 0.0);
         drawTriangle(2.0, 2.0, 3.0, 2.0, 2.5, 3.0, -10.0);
     glPopMatrix();
     glFlush();
     hits = glRenderMode(GL_RENDER);
     processHits(hits, selectBuf);
}
void init(void)
{
  glEnable(GL_DEPTH_TEST);
  glShadeModel(GL_FLAT);
}
void display(void)
{
  glClearColor(0.0, 0.0, 0.0, 0.0);
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   drawScene();
  selectObjects();
  glFlush();
}
int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
  glutInitWindowSize(200, 200);
  glutInitWindowPosition(100, 100);
  glutCreateWindow(argv[0]);
  init();
  glutDisplayFunc(display);
  glutMainLoop();
  return 0;
}
```

في هذا المثال لدينا أربع مثلثات (أخضر و أحمر و إثنين أصفرين تم إنشاءهما بإستدعاء الدالة drawTriangle و كل هذا في النمط الإفتراضي نمط التصيير و هناك مكعب سلكي wireframe box يمثل بدوره حجم الرؤية viewing volume ودالته drawViewVolume كلها ظاهرة على الشاشة. و بعدها نعيد تصيير المثلثات من جديد و هذه المرة نعيد تصيرها في نمط التحديد. التداخلات المتولدة يتم معالجتها في الدالة processHits و يتم عرض محتوى مصفوفة البيانات , فالمثلث الأول يولد تداخل واحد و و الثاني لا يولد شيئا و الثالث و الرابع يولدان نفس التداخل المشترك بنهما.

النقـــــر Picking

كإمداد لما وصفناه سابقا بإمكاننا إستعمال نمط التحديد لحساب ما إذا كان قد تم النقر على الكائنات , هذا سابقا أما الآن فسنستعمل مصفوفة نقر خاصة متحدة مع مصفوفة الإسقاط لنحصر مجال صغير من منفذ الرؤية view port و كل هذا يجب أن يكون وسطه مؤشر الفأرة بالنسبة لموقع للكاميرا,

تم تصغير الصورة بنسبة 85% (كانت 768 x 614) - انقر على الصورة لعرضها بحجمها الحقيقي



يتم إعداد النقر دائما بنفس الطريق مع بعض الإضافات التالية:
1- النقر عادة يدعم عن طريق دوال أو وسائل الإدخال input device , في المثال القادم يقوم المستخدم بالضغط على الجانب الأيسر للفأرة لإستدعاء الدالة المسؤولة عن النقر.
2- نستعمل روتين محدد من قبل المكتبة glu الدالة gluPickMatrix لضرب مصفوفة الإسقاط بمصفوفة النقر الخاصة و هذه الدالة يتم إستدعاءها دائما قبل إستدعاء دوال الإسقاط مثل gluPerspective أو glOrtho أو glFrustum أو .. , و من المحتمل أنك تريد حجز مصفوفة الإسقاط لذا فإتبع الشيفرة التالية:

```
glMatrixMode(GL_PROJECTION);
glPushMatrix();
glLoadIdentity();
gluPickMatrix(...);
gluPerspective, glOrtho, gluOrtho2D, or glFrustum
/* ... draw scene for picking ; perform picking ... */
glPopMatrix();
```

```
void gluPickMatrix(GLdouble x, GLdouble y, GLdouble width,GLdouble height, GLint viewport[4]);
```

هذه الدالة تقوم بإنشاء مصفوفة إسقاط و التي تحصر الرسم على بقعة صغيرة من منقذ الرؤية و ضرب المصفوفة في مكدس المصفوفة الحالية.
البارمترين الأول و الثاني x و y يمثلان مركز مكان النقر و هما إحداثيي النافذة و عادة يكونان هما نفسهما إحداثيي مؤشر الفأرة.
البارمترين width و height بمثلان طول و عرض بقعة أو مكان النقر في شاشة النافذة.
viewport هو منفذ الرؤية الحالي و نتحصل عليها بالإستدعاء التالي

```
glGetIntegerv(GL_VIEWPORT, GLint *viewport);
```

هذا لمثال لنقر

```
void init(void)
{
  glClearColor(0.0, 0.0, 0.0, 0.0);
  glEnable(GL_DEPTH_TEST);
```

```
  glShadeModel(GL_FLAT);
  glDepthRange(0.0, 1.0); /* The default z mapping */
}
void drawRects(GLenum mode)
{
  if (mode == GL_SELECT)
      glLoadName(1);
  glBegin(GL_QUADS);
    glColor3f(1.0, 1.0, 0.0);
    glVertex3i(2, 0, 0);
    glVertex3i(2, 6, 0);
    glVertex3i(6, 6, 0);
    glVertex3i(6, 0, 0);
  glEnd();
  if (mode == GL_SELECT)
       glLoadName(2);
  glBegin(GL_QUADS);
  glColor3f(0.0, 1.0, 1.0);
  glVertex3i(3, 2, -1);
  glVertex3i(3, 8, -1);
  glVertex3i(8, 8, -1);
  glVertex3i(8, 2, -1);
  glEnd();
  if (mode == GL_SELECT)
      glLoadName(3);
  glBegin(GL_QUADS);
  glColor3f(1.0, 0.0, 1.0);
  glVertex3i(0, 2, -2);
  glVertex3i(0, 7, -2);
  glVertex3i(5, 7, -2);
  glVertex3i(5, 2, -2);
  glEnd();
}
void processHits(GLint hits, GLuint buffer[])
{
  unsigned int i, j;
  GLuint names, *ptr;
  printf("hits = %d\n", hits);
  ptr = (GLuint *) buffer;
  for (i = 0; i < hits; i++) { /* for each hit */
   names = *ptr;
   printf(" number of names for hit = %d\n", names); ptr++;
   printf(" z1 is %g;", (float) *ptr/0x7fffffff); ptr++;
   printf(" z2 is %g\n", (float) *ptr/0x7fffffff); ptr++;
   printf(" the name is ");
   for (j = 0; j < names; j++) { /* for each name */
          printf("%d ", *ptr); ptr++;
   }
   printf("\n");
 }
}
#define BUFSIZE 512
void pickRects(int button, int state, int x, int y)
{
  GLuint selectBuf[BUFSIZE];
  GLint hits;
  GLint viewport[4];
  if (button != GLUT_LEFT_BUTTON || state != GLUT_DOWN)
          return;
  glGetIntegerv(GL_VIEWPORT, viewport);
  glSelectBuffer(BUFSIZE, selectBuf);
  (void) glRenderMode(GL_SELECT);
  glInitNames();
  glPushName(0);
    glMatrixMode(GL_PROJECTION);
    glPushMatrix();
      glLoadIdentity();
      /* create 5x5 pixel picking region near cursor location */
      gluPickMatrix((GLdouble) x, (GLdouble) (viewport[3] - y),
      5.0, 5.0, viewport);
      glOrtho(0.0, 8.0, 0.0, 8.0, -0.5, 2.5);
      drawRects(GL_SELECT);
    glPopMatrix();
    glFlush();
    hits = glRenderMode(GL_RENDER);
```

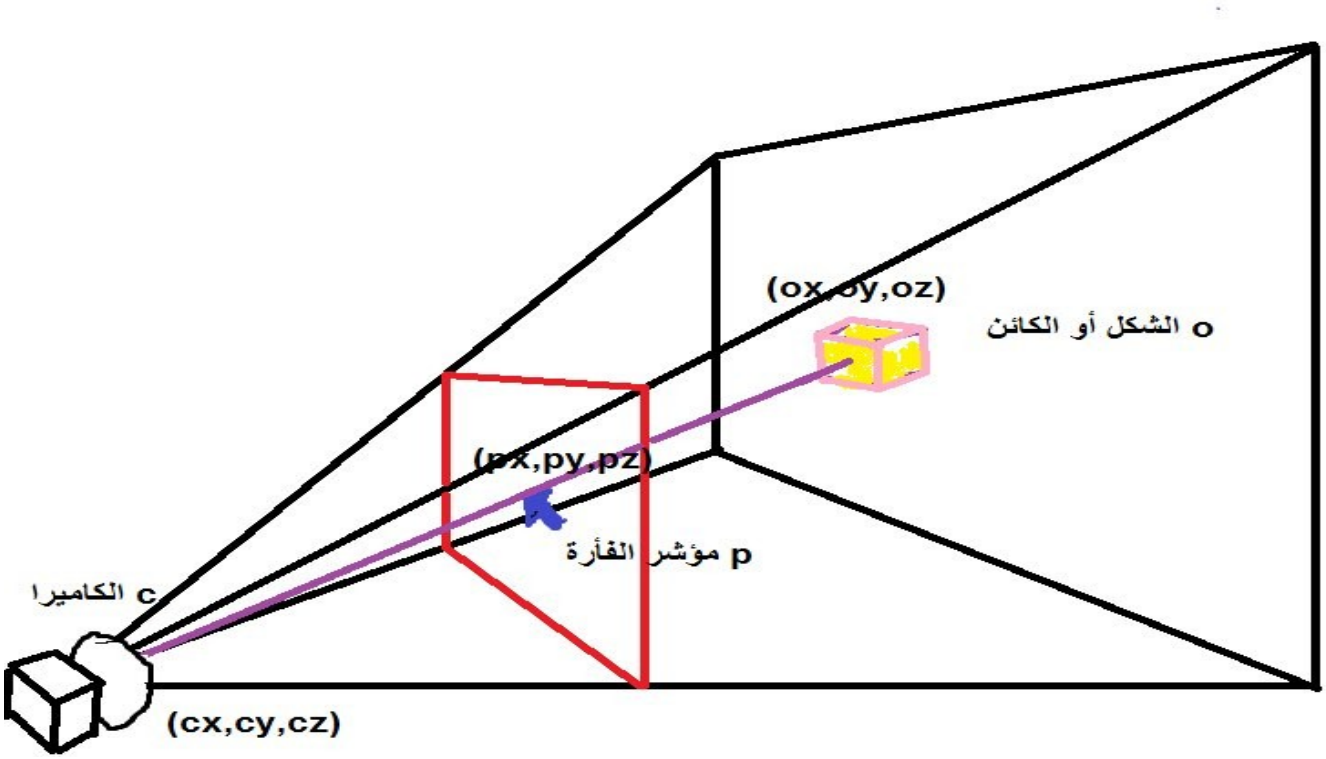
```
    processHits(hits, selectBuf);
}
void display(void)
{
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  drawRects(GL_RENDER);
  glFlush();
}
void reshape(int w, int h)
{
  glViewport(0, 0, (GLsizei) w, (GLsizei) h);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(0.0, 8.0, 0.0, 8.0, -0.5, 2.5);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
}
int main(int argc, char **argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
  glutInitWindowSize(200, 200);
  glutInitWindowPosition(100, 100);
  glutCreateWindow(argv[0]);
  init();
  glutMouseFunc(pickRects);
  glutReshapeFunc(reshape);
  glutDisplayFunc(display);
  glutMainLoop();
  return 0;
}
```

شرح الدالة processHits :
نمرر لهذه الدالة البارمترين hits و يمثل عدد التداخلات و البارميتر buffer يمثل مصفوفة البيانات المرجعة من قبل الأوبن جي أل.
نعرف المتغيرات i و j و names و *ptr
نجعل المتغير ptr يؤشر على المصفوفة (ليست هناك ضرورة لإستعمال المؤشرات إلا أن إستعمالها يزيد من سرعة التنفيذ) buffer
الدخول إلى أول حلقة من i يدل على أننا في التداخل الأول , الخانة الأولى من المصفوفة تحتوي على رقم يمثل عدد الأسماء التي تقاطعت مع حجم الرؤية, ثم يتم القفز غلى الخانة الثانية من المصفوفة بزيادة المؤشر ptr و بدورها تحتوي على العمق الأول و الخانة الثالثة في العمق الثاني و إبتداءا من الرابعة يوجد الأسماء فالخانة الرابعة يوجد الإسم الأول و الخانة الخامسة يوجد الإسم الثاني و في الخانة السادسة يوجد الإسم الثالث و هكذا دواليك إلى غاية آخر إسم , و من ثم يتم الإنتقال إلى التداخل الثاني و هكذا دواليك أيظا.

التمثيل في عالم الأوبن جي أل:

تفيذ هذه الخاصية في تمثيل الكائنات في عالم الأوبن جي أل , قد نرى هذه الوسيلة في البرامج الثلاثية الأبعادك blander و 3d studio max و cinima4d ووو . حيث نعتمد على مؤشر الفأرة لإنشاء نقاط أو مكعبات أو دوائر أو أشكال أخرى في عالم الأوبن جي أل الثلاثي الأبعاد .

تم تصغير الصورة بنسبة 85% (كانت 768 x 614) - انقر على الصورة لعرضها بحجمها الحقيقي



لنفرض أنه لدينا إحداثيات الكاميرا و إحداثيا مؤشر الفأرة و نريد الحصول على إحداثيات الكائن أو الشكل المرغوب رسمه في المشهد و يبعد عن عن الناظر (الكامير أصلا) بنسبة معينة , لنرا ذلك.

co = t*cp

و من ثم

|ox-cx| = |px-cx| l

|oy-cy| = t*|py-cy| l

|oz-cz| = |pz-cz| l

و من ثم

ox=t*(px-cx)+cx

oy=t*(py-cy)+cy

oz=t*(pz-cz)+cz

و بهذه الدالة نحصل على إحداثيات الشكل (ox,oy,oz)

بإمكانك التلاعب بالمعطيات لإعطاء الموقع المرغوب فيه للشكل (أضنك تفهم قصدي).

يمكن تمثيل موقع الكاميرا بإستخدام الدالة gluLookAt

```
void gluLookAt(GLdouble eyex, GLdouble eyey, GLdouble eyez,
GLdouble centerx, GLdouble centery, GLdouble centerz,
GLdouble upx, GLdouble upy, GLdouble upz);
```

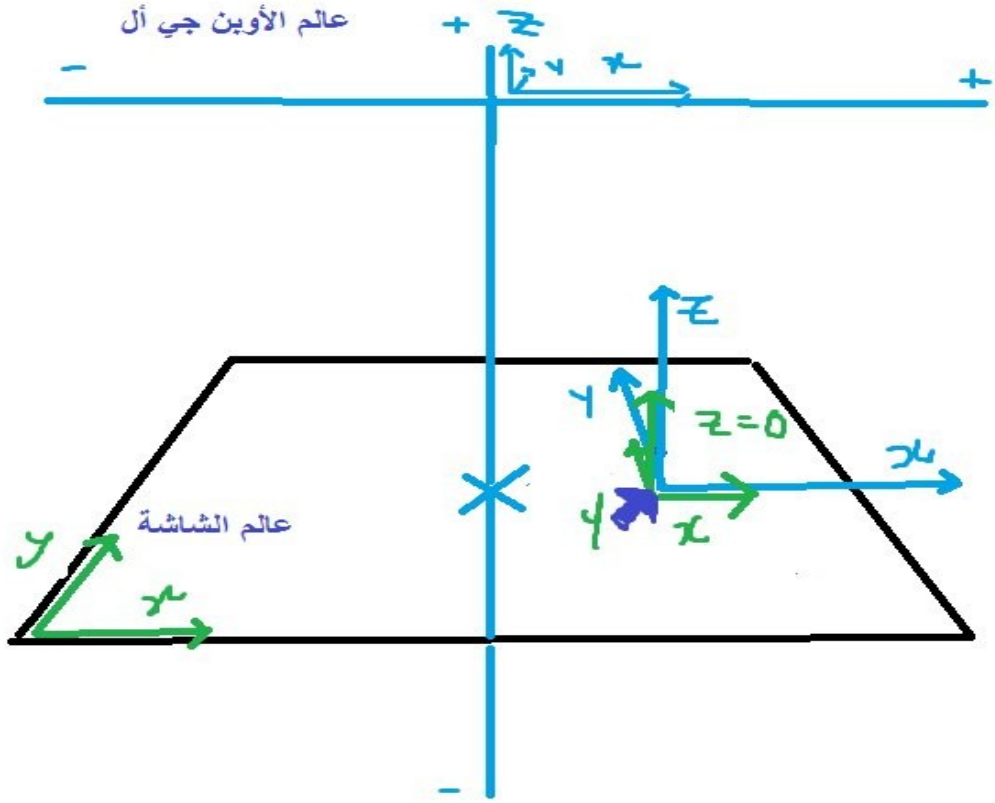
موقع الكاميرا الذي نعنيه و هو عين الناظر (eyex,eyey,eyez) و هو c .

أما التعقيب حول موقع مؤشر الفأرة في العالم الثلاثي الأبعاد فنستدعي الدالة gluUnProject

```
int gluUnProject(GLdouble winx, GLdouble winy, GLdouble winz,
const GLdouble modelMatrix[16],
const GLdouble projMatrix[16],
const GLint viewport[4],
GLdouble *objx, GLdouble *objy, GLdouble *objz);
```

هذه الدالة تقوم بإعطاء إحداثيات في العالم الأوبن جي أل لأي إحداثيات في نافذة الشاشة

تم تصغير الصورة بنسبة 85% (كانت 768 x 614) - انقر على الصورة لعرضها بحجمها الحقيقي



فالإحداثيات الممثلة بالأزرق هي إحداثيات عالم الأوبن جي أل أما الممثلة بالأخضر فهي للشاشة.
ففي الأخير نقول أننا نقدم للدالة gluUnProject إحداثيات الشاشة فتعطينا إحداثيات الأوبن جي أل.
و لعمل هذا نمرر لها أيظا مصفوفة الرؤية modelMatrix ومصفوفة الإسقاط projMatrix و مصفوفة منفذ الرؤية viewport
البارمترات GLdouble winx و GLdouble winy و GLdouble winz هي الإحداثيات في الشاشة و GLdouble objx* و GLdouble *objy و GLdouble *objz هي إحداثيات العالم الأوبن جي أل المرجعة .

هذا مثال نثبت فيه القيمة winz بصفر 0 و المعامل t=3 فيعطينا الموقع المرغوب.

```
#include <windows.h>
#include <GL/glut.h>
#include <stdio.h>
#define RED 1
GLdouble wx, wy, wz, Mx, My, Mz, Ax, Ay, Az, t, tab[10][3];
int ss = 0;
void display(void)
{
 glClear(GL_COLOR_BUFFER_BIT);
 glColor3f(1.,1.,1.);
     glPushMatrix();
    glTranslatef(0.,0.,-0.2);
    glutSolidSphere(0.1,10,10);
  glPopMatrix();

  glPushMatrix();
    glTranslatef((float)Mx,(float)My,(float)Mz);
    if (ss == 1)  glColor3f(0.,0.,1.);
            glutSolidSphere(0.01,10,10);
```

```
           glColor3f(1.,1.,1.);
            glPopMatrix();
  glFlush();
}
void reshape(int w, int h)
{
 glViewport (0, 0, (GLsizei) w, (GLsizei) h);
 glMatrixMode(GL_PROJECTION);
 glLoadIdentity();
 glFrustum(-0.5, 0.5, -0.5, 0.5,1,10);
 glMatrixMode(GL_MODELVIEW);
 glLoadIdentity();
 gluLookAt (Ax, Ay, Az, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
}
void mouse(int button, int state, int x, int y)
{
 GLint viewport[4];
 GLdouble mvmatrix[16], projmatrix[16];
 GLfloat realy;
   switch (button) {
   case GLUT_LEFT_BUTTON:
    if (state == GLUT_DOWN) {
      glGetIntegerv (GL_VIEWPORT, viewport);
      glGetDoublev (GL_MODELVIEW_MATRIX, mvmatrix);
      glGetDoublev (GL_PROJECTION_MATRIX, projmatrix);
      realy = viewport[3] - (GLint) y ;
      printf ("Coordinates at cursor are (%4d, %4d)\n", x, (int)realy);
        gluUnProject ((GLdouble) x, (GLdouble) realy, (GLdouble) 0,
mvmatrix, projmatrix, viewport, &wx, &wy, &wz);
      t=3;  // fix t to 3
      Mx = t*(wx-Ax)+Ax; //we calacul position of m with t
        My = t *(wy-Ay) +Ay;
      Mz =t*(wz-Az)+Az ;
       printf ("Coordinates at cursor are (%4f, %4f,%4f)\n", (float)Mx, (float)My, (float)Mz);
       glutPostRedisplay();
    }
    break;
   case GLUT_RIGHT_BUTTON:
    if (state == GLUT_DOWN)
      exit(0);
    break;
   default:
    break;
 }
}
void keyboard(unsigned char key, int x, int y)
{
 switch (key) {
   case 27:
    exit(0);
    break;
 }
}
void processMenuEvents(int option) {
switch (option) {
case RED :
ss=1;
break;
}
glutPostRedisplay();
}
int main(int argc, char** argv)
{
  wx=0; wy=0; wz=0;
  Ax=2; Ay=2; Az=5.0;
 glutInit(&argc, argv);
 glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
 glutInitWindowSize (500, 500);
 glutInitWindowPosition (100, 500);
 glutCreateWindow (argv[0]);
 int menu = glutCreateMenu(processMenuEvents);
 glutAddMenuEntry("Red",RED);
 glutAttachMenu(GLUT_RIGHT_BUTTON);
 glutDisplayFunc(display);
 glutReshapeFunc(reshape);
```

```
 glutKeyboardFunc (keyboard);
 glutMouseFunc(mouse);
 glutMainLoop();
 return 0;
}
```