Giovanni Baratta - 0000853633 - Laboratorio 1

Disegnare la curva di Bézier a partire dai punti di controllo inseriti, utilizzando l'evaluator di OpenGL

Per disegnare la curva con il valutatore interno di OpengL è stato sfruttata la funzione `glEvalCoord1f` valutando la curva in 100 punti differenti.

```
        glBegin(GL_LINE_STRIP);
        // valuto la curva in 100 punti e collego i punti successivi con dei segmenti
        for (i = 0; i < 100; i++)
            // valutazione e rendering della curva tramite opengl
            glEvalCoord1f((GLfloat)i / 100.0);
        glEnd();
```

Sostituire alle routine di OpenGL il disegno della curva mediante algoritmo di de Casteljau

In questo caso, è stato implementato l'algoritmo di De Casteljau. Sono state fatte delle interpolazioni sui punti di controllo fino ad ottenere un singolo punto. `tempPoints` contiene i punti di controllo, `tempNPoints` il numero di punti di controllo e `param` il valore scelto per l'interpolazione.

```
    while (tempNPoints > 1) {
        // faccio le interpolazioni sui segmenti
        for (i = 0; i < currentControlPoints - 1; i++) {
            tempPoints[i][0] = (1 - param) * tempPoints[i][0] + param * tempPoints[i + 1][0];
            tempPoints[i][1] = (1 - param) * tempPoints[i][1] + param * tempPoints[i + 1][1];
        }
        tempNPoints--;
    }
```

Come nel caso precedente, la curva è stata valutat in 100 punti diffenrenti.

```
        glBegin(GL_LINE_STRIP);
        // valuto la curva in 100 punti e collego i punti successivi con dei segmenti
        for (i = 0; i < 100; i++) {
            deCasteljau((float)i / 100.0, deCastResult);
            // rendering delle valutazioni
            glVertex3f(deCastResult[0], deCastResult[1], 0.0);
        }
        glEnd();
```

Disegno di una curva di Bézier mediante algoritmo ottimizzato basato sulla suddivisione adattiva

Per la suddiddivione adattiva è stato realizzato un algoritmo ricorsivo che controlla la distanza tra il segmento, formato dal primo e ultimo punto di controllo, e ogni punto di controllo della curva. Nel caso sia maggiore, la curva viene spezzata e si applica nuovamente l'algoritmo alle curve generate.

```
    for (int i = 1; i < numberOfControls - 1 && noSubdivision == false; i++) {
        x = controlPoints[i][0]; y = controlPoints[i][1]; z = controlPoints[i][2];
        distance =
            pointLineDistanceSquared(x, y, segmentStartX, segmentStartY, segmentEndX, segmentEndY);

        // se distanza > maggiore della soglia,
        //spezzo la curva in due parti e procedo ricorsivamente.
        if (distance > threshold) {
            noSubdivision = true;
            float leftCurve[MAX_NUM_PTS][3]; float rightCurve[MAX_NUM_PTS][3];
            divideCurve(controlPoints, numberOfControls, leftCurve, rightCurve);
            subdivisionCount += subdivision(threshold, leftCurve, numberOfControls)
                              + subdivision(threshold, rightCurve, numberOfControls)
                              + 2;
        }
    }
```

Per dividere la curva è stato utilizzato l'algoritmo di De Casteljau visto precedentemente, utilizzando un valore di `param = 0.5` e salvando i punti di controllo trovati durante l'interpolazione.

Permette la modifica della posizione dei punti di controllo tramite trascinamento del mouse

Per realizzare la funzionalità vengono sfruttare le callback `glutPassiveMotionFunc`, `glutMotionFunc` e `glutMouseFunc`. La prima è necessaria ad identificare il punto di controllo che si indende spostare. Ad ogni movimento viene verificata la distanza tra la posizione del mouse e i punti di controllo presenti, se la distanza è inferiore ad una soglia viene settato il valore di `mouseOverPointIndex` con l'indice del punto di controllo vicino al mouse.

```
    for (int i = 0; i < currentControlPoints; i++) {
        float distance =
            sqrt(
                pow(xPos - controlPoints[i][0], 2.0)
                + pow(yPos - controlPoints[i][1], 2.0)
                );
        if (distance < DISTANCE_THRESHOLD) {
            mouseOverPointIndex = i;
            break;
```

Nella callback `glutMouseFunc` viene identificato il click del pulstante sinistro, se `mouseOverPointIndex` contiene un indice valido (≥ 0), il valore della variabile `dragging` viene settato a `true`. Infine in `glutMotionFunc`, se `draggin` ha valore `true`, viene aggiornata la posizione del punto di controllo con indice `mouseOverPointIndex`. Alla display successiva il punto di controllo verrà disegnato nella nuova posizione.

```
    if (dragging && mouseOverPointIndex >= 0) {
        // sposto il punto
        controlPoints[mouseOverPointIndex][0] = xPos;
        controlPoints[mouseOverPointIndex][1] = yPos;
        glutPostRedisplay();
    }
```