Practica 1
Informàtica Gràfica

Jaume Pla Ferriol - *u1941090*

20 d'octubre de 2024

1 Idea inicial

El meu objectiu principal en plantejar aquesta pràctica va ser fer un projecte prou senzill com per a no sobrecomplicar l'escena, poder-me enfocar en aprendre com funciona l'ecosistema WebGL, i utilitzar tots els elements requerits per a la pràctica. Vaig decidir-me en fer un "generador de constel·lacions": un fons d'estrelles senzilles fetes a partir de POINTS (*background*), estrelles instanciades per l'usuari, de número de puntes variable, fetes amb TRIANGLE_FAN, i unes línies que ajuntessin aquestes estrelles, fetes amb LINE_STRIP (*foreground*), i amb l'opció de poder fer línies noves.

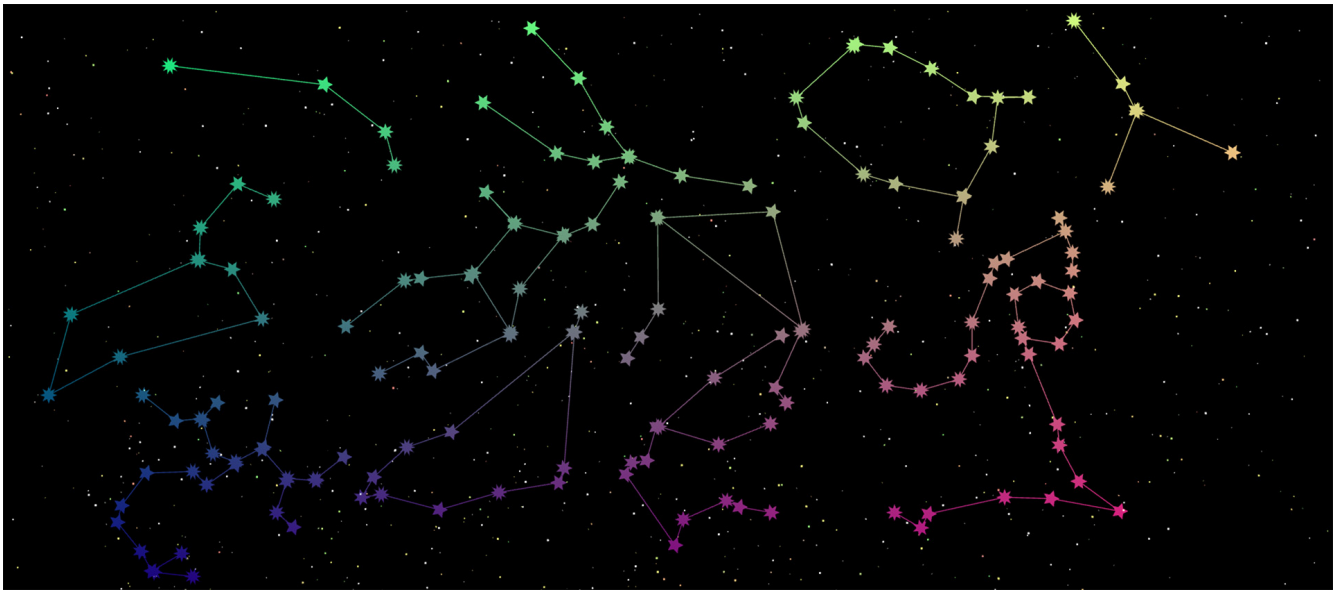

Figure 1: Constel·lacions dibuixades amb l'eina

2 Funcions per generar punts

- **Cercle:** La funció *polyCircle(segments, radius, center, fill)* crea els punts d'un cercle de cares *segments*, radi *radius* i punt central *center*. També té la opció de fer *fill*: repetir el punt[1], per poder dibuixar bé el cercle amb TRIANGLE_FAN. Per a aquesta pràctica només s'utilitza aquesta funció com a eina per a generar els punts de les estrelles del *foreground*, així que la opció de *fill* no s'utilitza.

- **Estrella:** La funció *polyStar(sides, r1, r2, center)* crea els punts d'una estrella de *sides* costats, radi interior *r1*, radi exterior *r2* i punt central *center*. Es creen els punts de dos polígons idèntics i concèntrics, però de mides diferents. Modifiquem els índexs d'aquests punts (afegim *2*sides* als índexs dels punts *innerCircle*) i combinem els punts dels dos *arrays* de punts inicials alternament per a obtenir un *array starPoints* amb els punts necessaris per a dibuixar una estrella amb TRIANGLE_FAN.

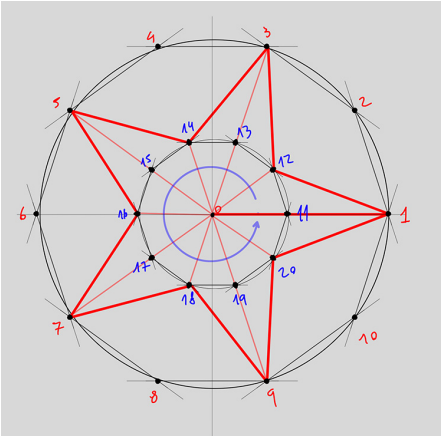

Figure 2: Esbós inicial

- **Punts escampats:** La funció *randPoints(pNum, maxSize, minSize)* genera *pNum* punts escampats en *clip space*, i els assigna un *VertexSize* aleatori entre *minSize* i *maxSize*.

3 Primitives

Per aquesta pràctica només he utilitzat un sol *shader*, així que he hagut de crear *buffers* per a diferents elements (*vertices, sizes, vCenter, t*) per a tots els modes de dibuixat, encara que no fossin necessaris per a cada un.

- **POINTS** - *drawPoints()*
  Les estrelles del *background* es dibuixen amb aquest mode, des de *shader* es controla *gl_PointSize* i color.

- **LINE_STRIP** - *drawLineStrip()*
  Les línies que ajunten les estrelles de cada constel·lació es dibuixen amb aquest mode. Per a aconseguir dibuixar diferents línies amb aquest mode, es creen dos objectes: *currentLine* guarda les dades de la línia que s'està dibuixant en aquell moment, i *lineArray* guarda totes les línies ja completades.

- **TRIANGLE_FAN** - *drawTriangleFan()*
  Les estrelles del *foreground* es dibuixen amb aquest mode. Per a aconseguir dibuixar diferents estrelles amb aquest mode, es crea un objecte *starArray*, que guarda totes les estrelles instanciades.

4 *Input* de l'usuari

- ***Mouse & keyboard***
  Left click: crea una nova estrella.
  Middle click: Començar una nova línia.
  Spacebar: comença una nova línia.

- ***Sliders***
  Background stars number: quantitat d'estrelles al *background*. Refresca l'escena.

Background star max size: mida màxima de les estrelles del *background*. Refresca l'escena.
Star scaling: mida de les estrelles del *foreground*. És una operació d'escalat per *shader*.

- **Botons**
  Clear: esborra tot el *foreground*.
  New Line: comença una nova línia.

5 Animació

Al crear una estrella s'inicia una funció asíncrona *timeline(model, steps)*, que actualitza *model.t* durant *steps* fotogrames. Per simplicitat en aquesta pràctica ho he deixat en 100 fotogrames. Aquest valor t es crida des del *vertex shader* com a *TransitionTime*, i mou els vèrtex de l'estrella des del seu centre.

6 Shader

Background
Els punts s'escalen a partir del seu valor *VertexSize*, i sel's assigna un color aleatòri, utilitzant la seva posició *VertexPosition* com a *seed*. S'utilitzen les variables *colorDesaturate* i *colorIntensity* per a controlar el color final.

Foreground
Per a assignar un color dels elements del *foreground* he utilitzat la seva posició en *clip space* per a aconseguir un degradat de colors *pastel* de forma molt fàcil, que s'adeqüen a l'estil de la resta d'elements de l'escena.
S'utilitza l'*input TransitionTime* per a fer una interpolació lineal entre les posicions de vèrtex *StarCenter* i *finalPos*.

$$\vec{\text{final}}\text{Pos} = (\text{Vertex}\vec{\text{P}}\text{osition} - \text{Star}\vec{\text{C}}\text{enter}) \times sScale + \text{Star}\vec{\text{C}}\text{enter} \quad (1)$$

El valor de *StarCenter* == 5.0 s'utilitza com a marcador de que aquell vèrtex pertany a una línia, i no s'hauria de modificar el seu *VertexPosition*.

7 Estructura del projecte

El projecte compta amb un document HTML *p1_main_HTML.html*, un script principal *p1_script.js*, un document de funcions de WebGL *p1_WebGL.js*, un document de funcions per generar punts *jShapeLib.js*, un document de funcions d'utilitat *jUtilLib.js* i un document d'estil *p1_style.css*. Ho he separat en aquests arxius per a mantenir el projecte més estructurat, i per a poder reutilitzar algunes parts d'aquest projecte en el futur de forma més senzilla.

8 Conclusions

Amb aquesta pràctica he entès bastant més com crear i gestionar *buffers* i *draw calls*. La part de *shaders* la coneixia bastant més, degut a la meva experiència com a *Technical Artist*, però ha estat interessant veure com implementar diferents primitives amb un sol shader.
He intentat mantenir un *look* coherent i minimalista, i crec que el resultat final ha acabat sent el que m'havia plantejat des d'un inici.