Implement Qt3D backend for KStars

Introduction

A planetarium program simulates the night sky, at any time of the day from any location of the earth. It has several applications for students, teachers or astronomers. They show the night sky using a sky map representation or a complex photorealistic sky view.

There are various planetarium programs available. The popular open-source ones being Stellarium, KStars and Sky Map. KStars is a project by KDE based on the Qt framework. It is available for Mac OSX, Linux and Windows. There also exists an Android version named KStars Lite.

Softwares like Stellarium render the night sky in 3D and use a photorealistic pipeline making it look as attractive as possible. On the other hand, SkyMap(formerly Google Sky Map) is available only for the Android platform and utilizes OpenGLES for its Sky Render. KStars however, utilizes another framework named QPainter to render the SkyMap using 2D drawings. This essentially becomes our problem statement for the project. Since there is a lack of a 3D backend to render all the sky objects(planets, stars, etcetera) on the skymap, the Skymap is simply not realistic enough.

Project Goals

Adding the Qt3D backend to KStars can be broken down into 3 parts.

- Create Qt3D based backend to draw all objects currently implemented by QPainter backend.

- Create realistic colors, shaders, textures, meshes, lighting for all-stars, solar system, and deep-sky objects.

- Create animations for meteor shows, comet tails, stars twinkle..etc

Implementation

I'll divide the implementation based on the goals. I'll also discuss the files which will be changed/added in the implementation.

- **Writing a Qt3D based backend.**

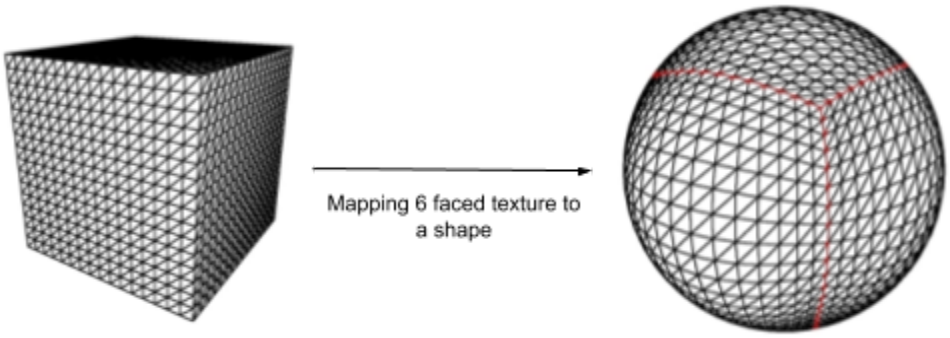
  The SkyPainter could utilize any of the backends it is provided with. Currently, we have a working implementation of SkyQPainter which draws all the SkyComponents in 2D.

  Firstly, we create a Qt3D based backend to add another implementation of the SkyPainter class. We look into the implementation of the following functions :

  - drawPlanet
  - drawDeepSkyObject
  - drawPointSource
  - drawSkyPolygon
  - drawSkyPolyline
  - drawSkyLine
  - drawSkyBackground
  - drawObservingList
  - drawFlags
  - end
  - begin
  - setBrush
  - setPen
  - drawHorizon
  - drawSatellite
  - drawSupernova
  - drawConstellationArtImage
  - drawHips

  Each of these functions will be implemented using appropriate texture mapping.

  For instance, planets(example Earth) could be implemented using CubeMap Tessellation. Here we generate a sphere from a cube and then an ellipsoid from the sphere by multiplying using WGS84 radius. The CubeMap holds a 6 faced texture of the planet. We could also use an acceleration structure like QuadTree or Octree.



Next, we implement the SkyMapDrawAbstract class for Qt3D. This would mean implementing the paintEvent() method in the new subclass. We also include the initialization and resize functions for Qt3D.

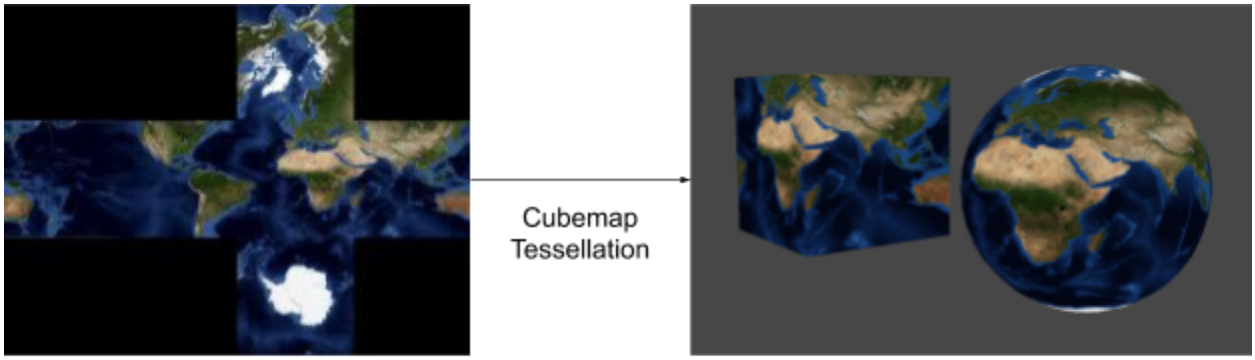
- **Adding realistic colors, shaders, textures and appropriate lighting for all SkyComponents.**

  For textures, we utilize imagery tiles. It can be done as follows :

  1. Finding the imagery tiles required for the component.
  2. Loading it as textures.
  3. Map it on geometric tile's vertices. For this, we could utilize a QBuffer on GPU which could be accessed by a shader.

  Our shaders then use indices to lookup for each tile in the QBuffer used. This could account for terrain elevations as well.

  We plan to utilize a point light-based system for the glowing celestial bodies. Qt3D provides us the QPointLight Class to achieve that.



6 faced cube texture for earth

- **Adding animations for events like meteor shows, comet tails, etcetera.**

  The current QPainter version doesn't have any animations. With Qt3D, we intend to add the following animations as well.

  - Meteor shows - We intend to use the animation concepts explained below to display a meteor shower for the particular date.
  - Comet Tails - We could use a QKeyFrame clip for animating a comet tail.
  - Atmospheric Transitions - Animations for day/night transitions.
  - Animations for Satellites and other bodies.

  Qt3D based animations could be done using 2 simple concepts:

  - Animation Data - Here we use an Animation Clip Loader to generate a QKeyFrame using the 3D model of the object. We achieve a simple keyframe based playback using Qt3D's Clip Animator.
  - Animation Targets - For reusing animation data in bodies like meteors, we depend on mapping the Animation data to multiple properties of multiple objects. For example, we can make animation targets for rotation and translation for 2 objects simultaneously and much more using the Channel Mapper.

  Since we are dealing with only celestial objects, there are no complex skeletal animations to be dealt with and we can come up with some basic good looking animations in the limited time.

Timeline

February - April

- For now, the boilerplate code which adds two additional classes, Sky3DPainter and SkyMap3DDraw has been written. These are supposed to implement SkyPainter and SkyMap3DDrawAbstract respectively.

- SkyMap3DDraw successfully paints the QWidget using Sky3DPainter when instantiated in SkyMap.

- Sky3DPainter only implements DrawSkyBackground for now.

- Documentation for both classes(taken from SkyQPainter and SkyMapQDraw).

May 04 - June 01 (Community Bonding Period)

- Interact and consult with my mentor.

- Dig deep into the implementation details as written in the proposal. Look for more optimum methods to do the project.

- Continuing with the documentation of the backend classes.

- Planning and formatting for future weekly reports.

- Setup the blog for Planet KDE and weekly posts.

- Bonding with the KDE community.

June 01 - June 15 (2 weeks)

- Implement the remaining SkyMap3DDraw for resize events and do necessary changes for paint events.

- Collecting textures for models.

- Planning how to approach rendering of each sky object in 3D.

- Implement the Sky3DPainter for deep-sky objects and the horizon(if not done before community bonding period.)

- Blog posts introducing new Qt3D backend.

June 15 - July 13 (4 weeks)

- Implement Sky3DPainter for the remaining sky objects.
- Write documentation for textures and lighting changes on each sky object.
- Applying correct textures, mesh construction, shaders, and lightings for various sky objects in Sky3DPainter.
- Review of Sky3DPainter class.
- Unit tests for each sky object and the Sky3DPainter class.
- Blog posts on implementation of Qt3D backend.

July 13 - August 03 (3 weeks)

- Planning for how to approach animations in the remaining time for the sky objects.
- Write documentation for animating some celestial bodies.
- Implementation of animations for selective celestial bodies as described in the proposal.
- Unit tests for Sky3DPainter again after adding animations.
- Blog posts covering animations in KStars.

August 03 - August 17 (2 weeks)

- Unit Testing for widgets and other options in KStars which failed with the GL backend earlier.
- Documentation for switching backends.
- Adding options for switching between QPainter and Qt3D.
- Blog posts on switching backends in KStars.

August 17 - August 24 (1 week)

- Testing for Sky3DPainter and bug fixes.
- Generate and check Doxygen documentation which was written earlier.
- Code clean up.
- Lookup into optimization techniques like Vertex and Depth precision.
- Blog post on optimizations in Qt3D based backend.

August 24 - August 31 (1 week)

- Get reviews from the mentor for overall activity
- Work on any desired changes.
- Final Submission and Evaluation.
- Blog post on final submission and overall experience.

Availability

I'll be dedicating 30 to 40 hours a week. My semester exams end in May first week and my internship contract at Laboratoire d'informatique de Grenoble expires on June 5th and I will be able to work more flexibly after that.

In any case, I'll report to the mentor and will always be available on IRC and Email.

About me

I am a Master's student at Université Grenoble Alpes and École nationale supérieure d'informatique et de mathématiques appliquées de Grenoble(ENSIMAG). My area of specialization in Graphics, Vision and Robotics. I am actively working on projects related to OpenGL, SciPy, Pytorch, etcetera.

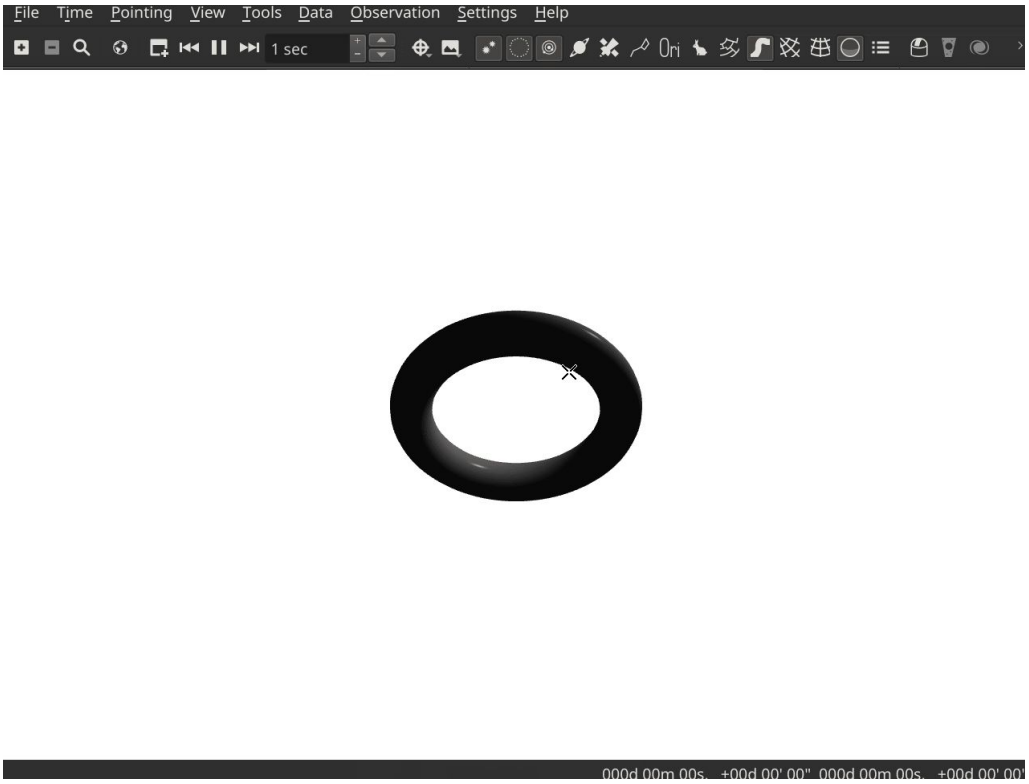
I have participated in GSoC in 2018 for Godot Engine where I worked on Native Engine Integration for Gear VR. I also wrote the GDNative API bindings for Godot core.

I have been doing open source contributions since 2018. Currently, I am actively contributing to KDE and another project named GazePlay.

Currently, I am a research intern at Laboratoire d'informatique de Grenoble. Where I am part of team GETALP and work on the eye-tracking interaction mechanisms for cognitively disabled people.

Apart from college, I play football and chess and love cooking Indian and French food.

Recently, I fixed a bug in KStars for correct equinox and solstice calculations in the calculator tool (https://phabricator.kde.org/D27413). I also implemented a sample Qt3D scene in the KStars Skymap to display how a 3D torus could be rendered, it is available on my fork.



Contact Info:
IRC: @paritosh97:kde.org
E-mail: paritosh.sharmas@gmail.com
Phabricator: https://phabricator.kde.org/p/paritosh/
GitHub: https://github.com/Paritosh97/