Usar o opengl no xamarin é mto ruim:

1. Existem conflitos de dll dentro do próprio xamarin que fazem com que várias funções não estejam disponíveis. Só que vc só descobre que essas fns n estão disponíveis no runtime. Por causa desses problemas crescentes eu só consegui ir até o ponto onde tinha que setar a textura: nesse ponto a quantidade de erros se acumulou a tal ponto que eu desisti.
2. Minha experiência com o opengl no xamarin foi muito muito ruim. O uso da opentk como encapsuladora do opengl foi um erro, na minha opinião.
3. O melhor jeito de usar seria escrever o código de opengl no projeto android e no projeto ios e manter só o modelo no projeto compartilhado. Só que se for pra fazer isso eu escrevo nativo, no android studio ou no xcode, pq a ideia seria justamente fazer o opengl multiplataforma

O uso da urhosharp

1. A documentação é péssima, não ensina como montar o projeto inicial.
2. Fui incapaz de montar um projeto que não crashasse com erros na camada nativa, dentro da JNI no caso do android. Não encontrei informações sobre isso em lugar algum da internet.
3. A solução que eu encontrei no final foi copiar a solução que funciona, que vem com os exemplos, renomeá-lo e compilar para testar. Com isso eu consegui algo, no caso, a compilação funcionar.

Estudando o urhosharp

1. Pra conseguir usar, copiei o projeto que demonstra o uso do urhoSharp.forms, o Charts. Só assim para não crashar nesse momento.
   1. Copiei a solução pra um diretório próprio, renomeei os projetos e os religuei. O projeto compartilhado é do tipo Portable Code Library.
   2. Dentro do projeto, na classe App, o layout com o componente do urho é criado manualmente, o UrhoPage criado ali, descendente de ContentPage.
   3. No OnAppearing() eu invoco o método que cria a “application”do urho.
   4. Essa aplication é onde a renderização é feita. E era aqui que crashava no c++ do android quando eu fazia as tentativas anteriores. Isso não era explicado de forma alguma na documentação da urhosharp no xamarin e só descobri copiando os exemplos e modificando-o.
      1. Hipótese: Nunca houve problema em usar .net core library como estratégia de compartilhamento. O problema é que eu não estava atendendo as condições prévias, mostradas no exemplo.
   5. No app implementei os métodos e criei o textinho na tela, fazendo o primeiro programa finalmente funcionar.
2. Criar um Cubo
3. Mexer na câmera
4. Textura do cubo

A experiência com o UrhoSharp por enquanto está muito ruim, pela falta de documentação (uma característica em comum dessas partes menos usadas do Xamarin).

Comparado com tecnologias alternativas, como a Unity, o xamarin para 3d está se revelando muito ruim por problemas de tooling e de falta de documentação (em particular posts de stack overflow sobre solução para problemas).

Agora que consegui fazer rodar algo pela primeira vez, uma versão copiada do charts do exemplo, é hora de eu fazer meu projeto.

TAREFAS

1. Fazer um projeto que rode.
   1. ~~Copiei e colei o projeto do Urho.forms.~~
   2. ~~Fiz com que o projeto já começasse usando o UrhoPage ao invés da página com o botão inicial.~~
   3. ~~Renomear os diretórios dos projetos.~~
2. Limpar o código do exemplo.
   1. ~~Criar um branch com o código nesse estado~~
   2. ~~Remover a chart~~
   3. ~~Remover os eventos da gui~~
   4. ~~Fazer o hello world (o texto na tela)~~
3. Criar um cubo
   1. Por o cubo na tela
   2. Por uma textura nele
   3. Fazer um shader customizado (se não der mto trabalho)
4. Fazer picking desse cubo
5. Controle de câmera
6. Arrastar o objeto na tela

Com isso eu terei algo para mostrar

Se der tempo:

1. Um websocket que receba posições de usuários
2. Dockerizar esse websocket
3. Varios clientes enviando suas posições.
4. O cliente puxar a posição dos outros clientes do servidor via websocket e atualizar a cena