

USO DA REALIDADE AUMENTADA COM MARCADORES DINÂMICOS

Aluno(a): Everton da Silva

Orientador: Dalton Solano dos Reis

Roteiro

- Introdução
- Objetivos
- Fundamentação teórica
- Requisitos
- Especificação
- Implementação
- Resultados
- Conclusão e sugestão
- Demonstração

Introdução

- Interação Homem-Máquina
- Realidade Aumentada
- Marcadores

Objetivos

- **Objetivos específicos**
 - Importar um arquivo 3D (.fbx)
 - Mover e rotacionar na posição desejada
 - Gerar bordas do objeto selecionado
 - Sobrepor as bordas na tela do dispositivo
 - Detectar desenho feito pelo usuário com o objeto selecionado
 - Sobrepor objeto 3D sobre o desenho

Fundamentação Teórica

- OpenCV
- OpenCV For Unity (Enox Software)
- Homografia E Transformação De Perspectiva
- Filtro De Canny
- Transformação Morfológica

Trabalhos Correlatos

- iAR
- ANIMAR
- Estudo e implementação de técnicas de realidade aumentada

Requisitos Funcionais

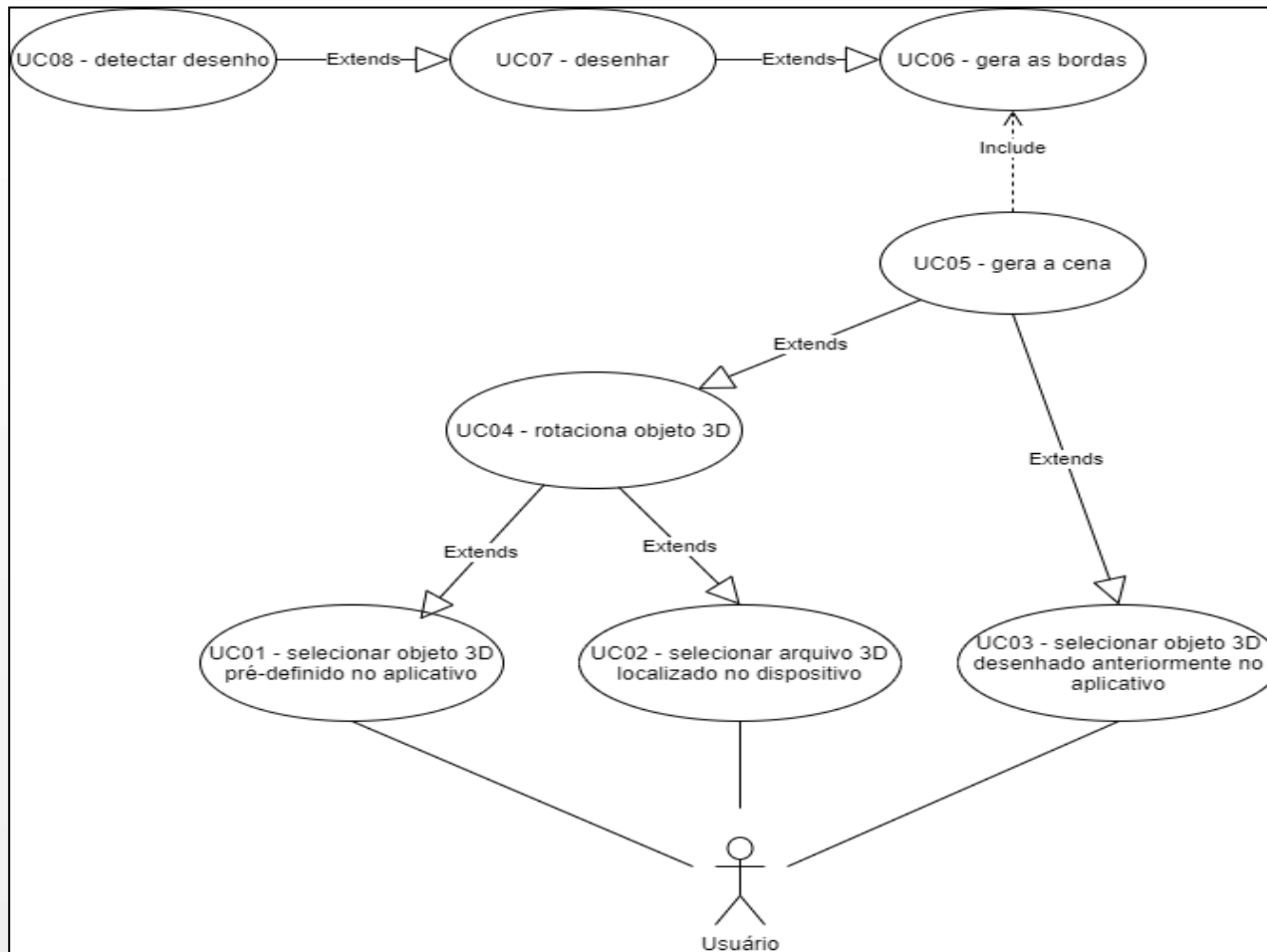
- permitir selecionar um objeto predefinido no aplicativo
- permitir selecionar um arquivo localizado no aplicativo
- permitir selecionar um objeto já desenhado anteriormente
- permitir rotacionar um objeto na tela do dispositivo
- permitir gerar cena
- gerar as bordas do objeto selecionado
- sobrepor na tela do dispositivo as bordas do objeto selecionado
- comparar desenho do usuário com bordas do objeto selecionado

Requisitos Não Funcionais

- ser desenvolvida para iOS e Android
- utilizar o ambiente Unity para desenvolvimento
- utilizar as bibliotecas ArUco para detectar marcadores
- utilizar biblioteca OpenCV para Unity para reconhecimento de imagens
- utilizar recurso Unity para salvar informações no dispositivo

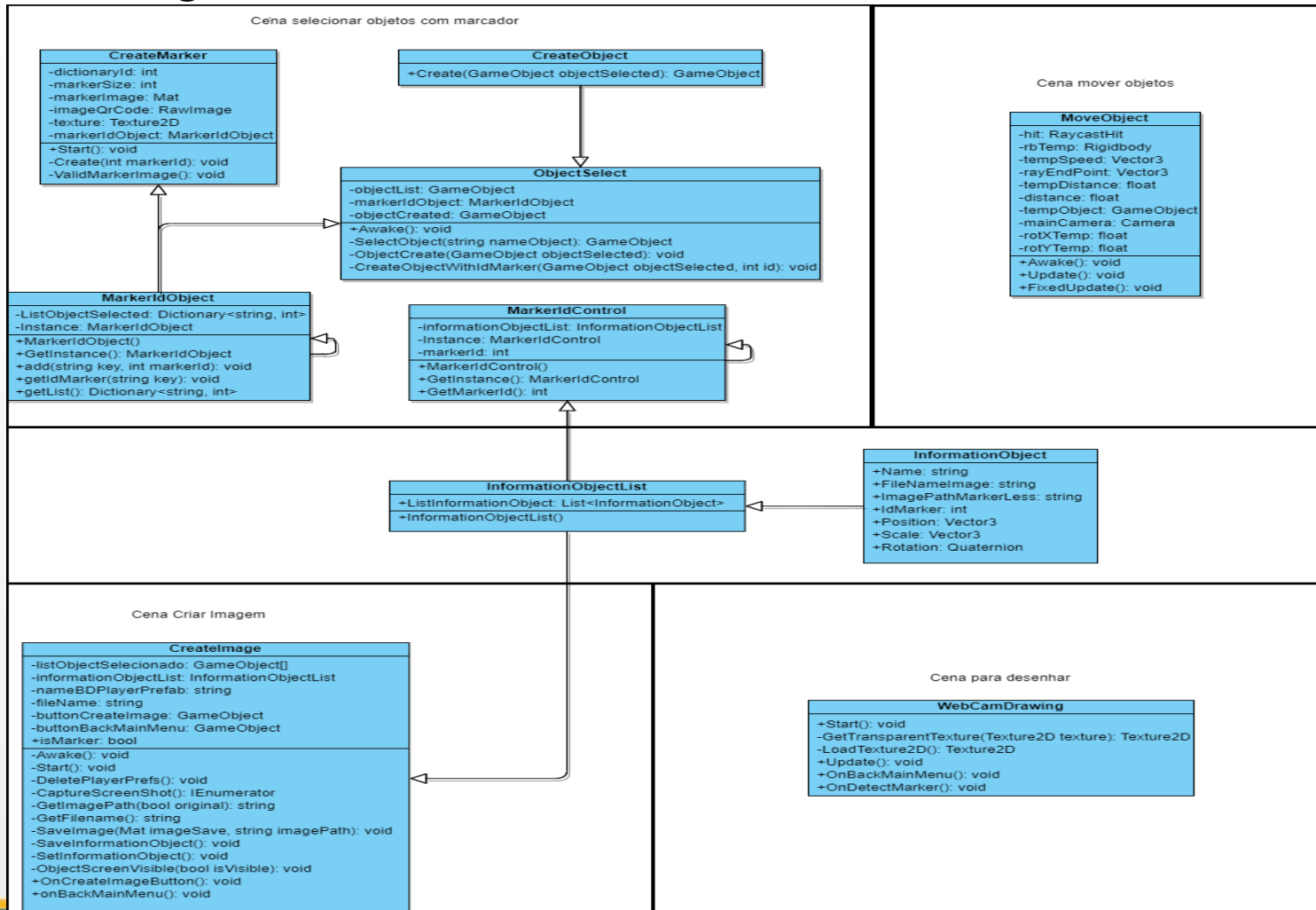
Especificação

- Diagrama de Casos de Uso



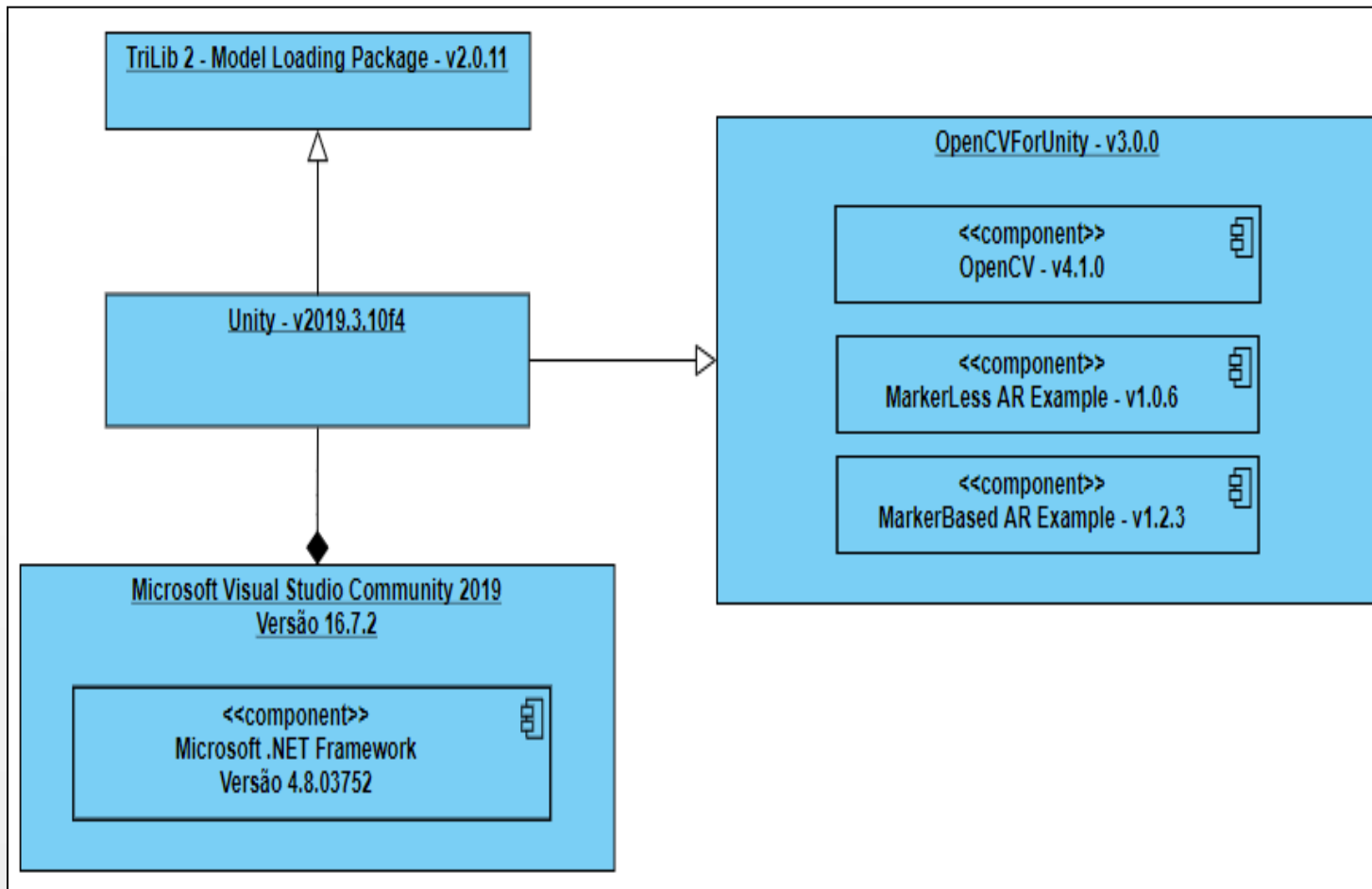
Especificação

- Diagrama de Classes



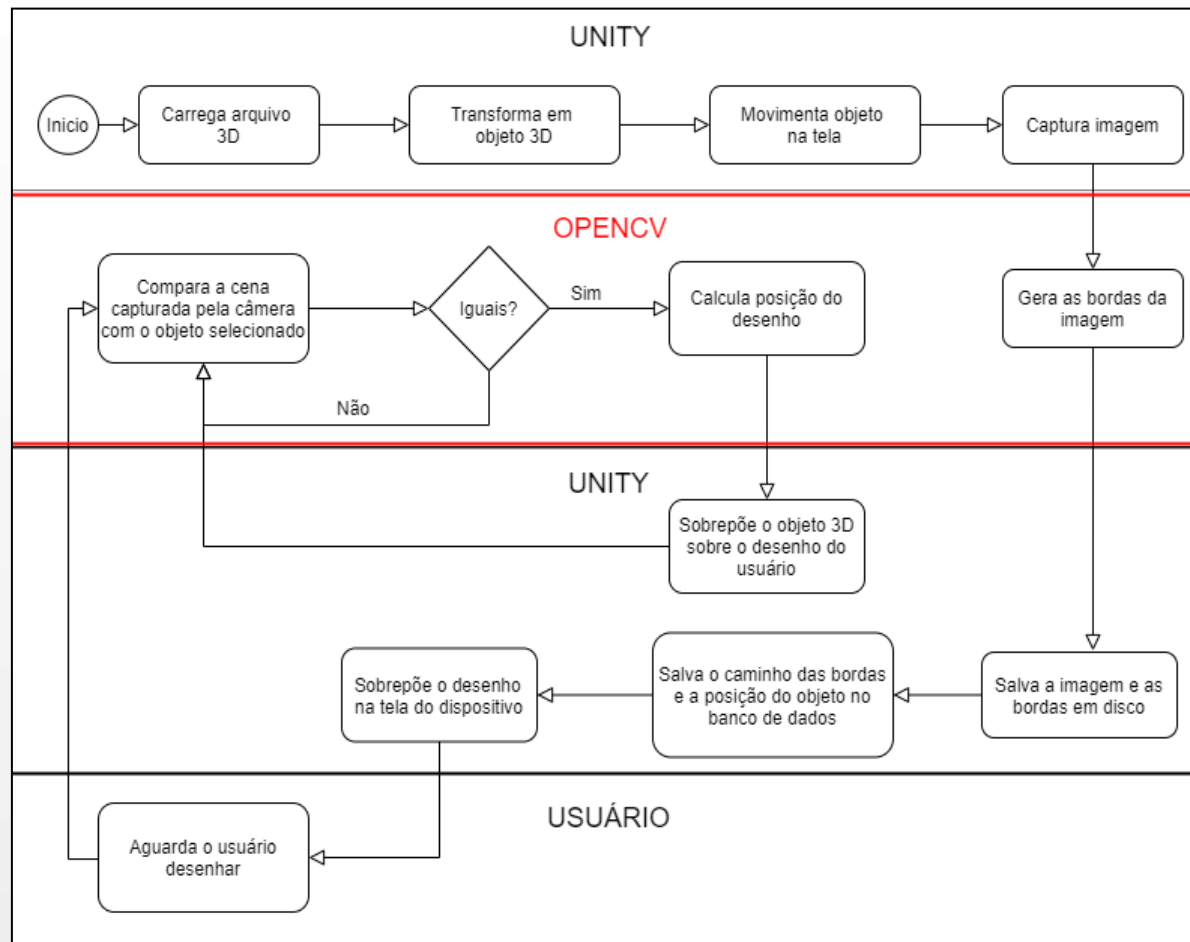
Especificação

- Diagrama de arquitetura



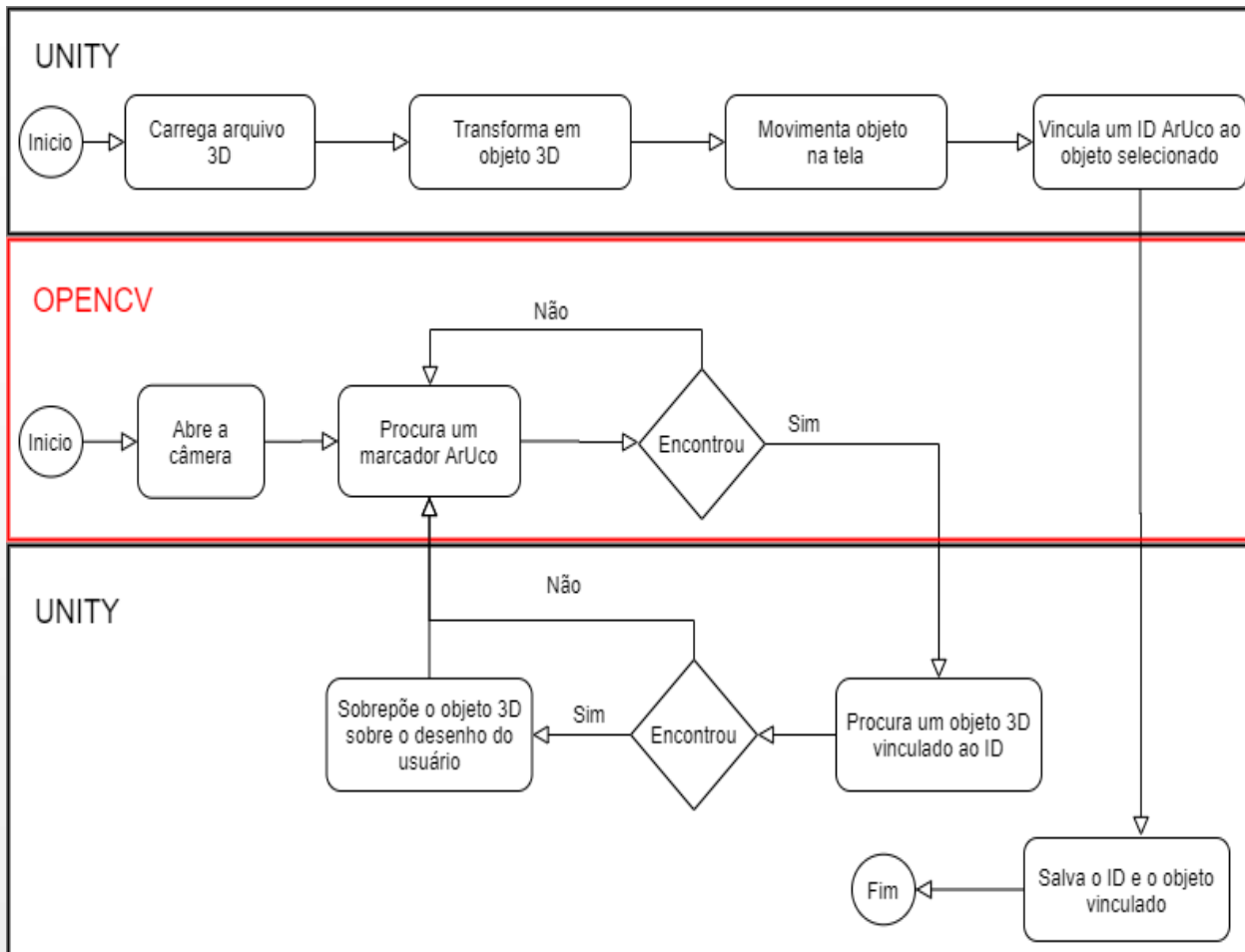
Especificação

- Diagrama de atividade sem marcador predefinido



Especificação

- Diagrama de atividade com marcador predefinido

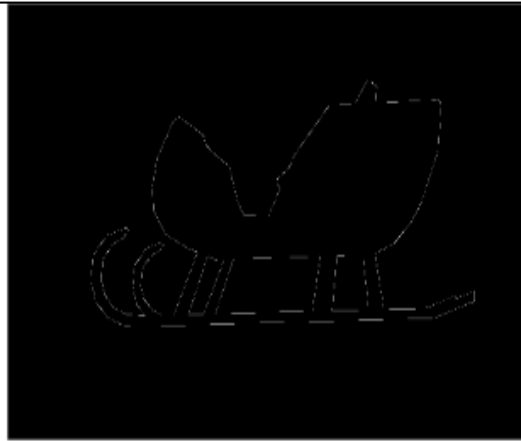


Implementação

- Gerar bordas



1) Imagem original



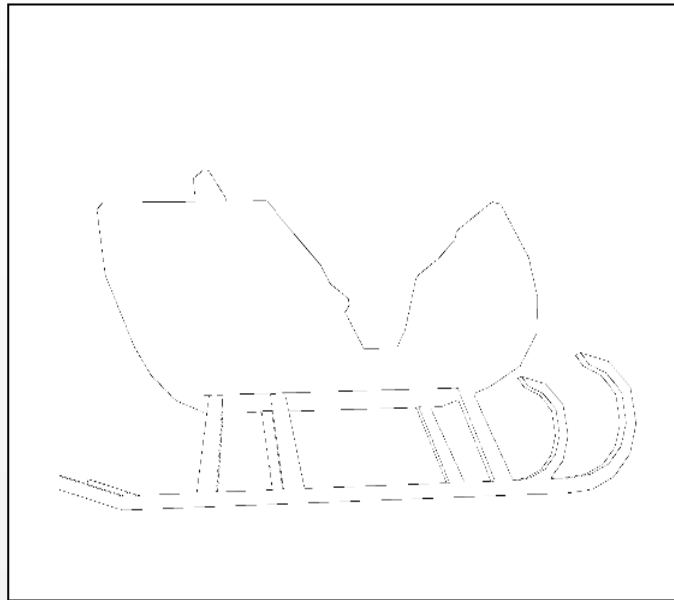
2) Filtro de Canny



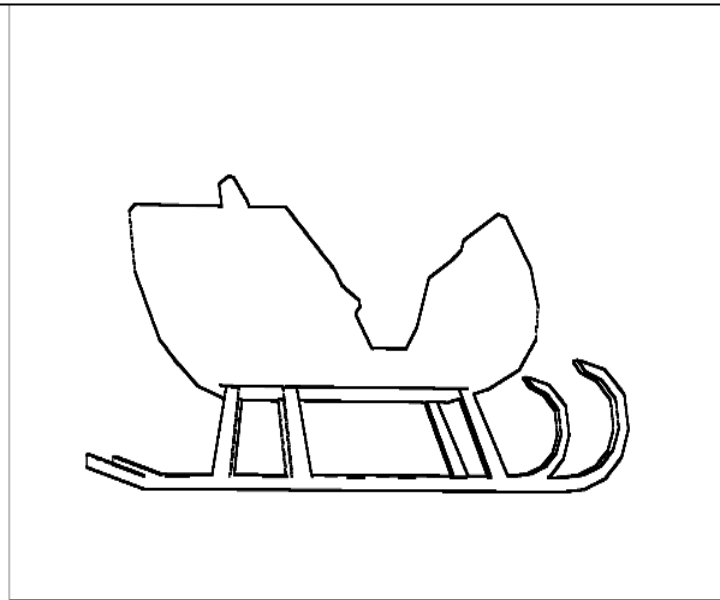
3) Método bitwise_not

Implementação

- Dilatação



1) Imagem sem dilatação



2) Imagem após o processo de dilatação

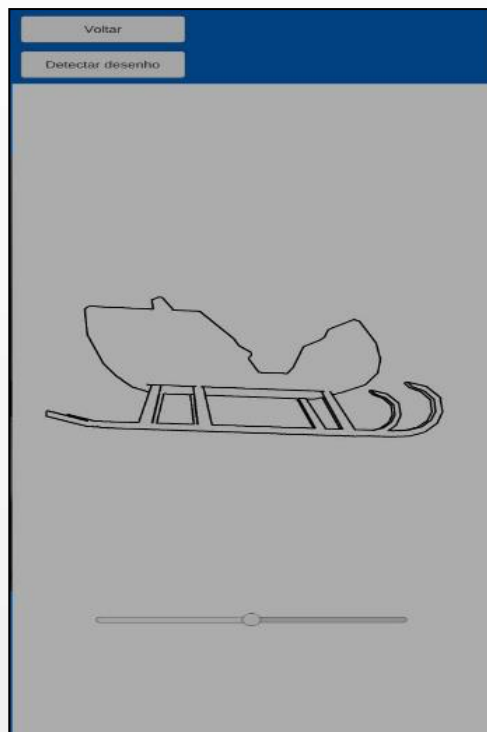
Implementação

- Método GetTransparentTexture

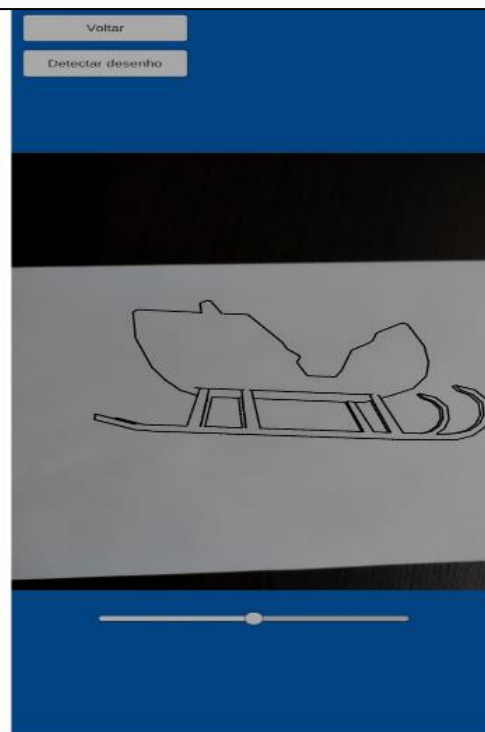
```
150 public Texture2D GetTransparentTexture(Texture2D texture)
151 {
152     Color transparentColor = new Color(1.0f, 1.0f, 1.0f, 0f);
153
154     for (int y = 0; y < texture.height; y++)
155     {
156         for (int x = 0; x < texture.width; x++)
157         {
158             if (!Color.black.Equals(texture.GetPixel(x, y)))
159             {
160                 texture.SetPixel(x, y, transparentColor);
161             }
162         }
163     }
164
165     texture.Apply();
166     return texture;
167 }
```


Implementação

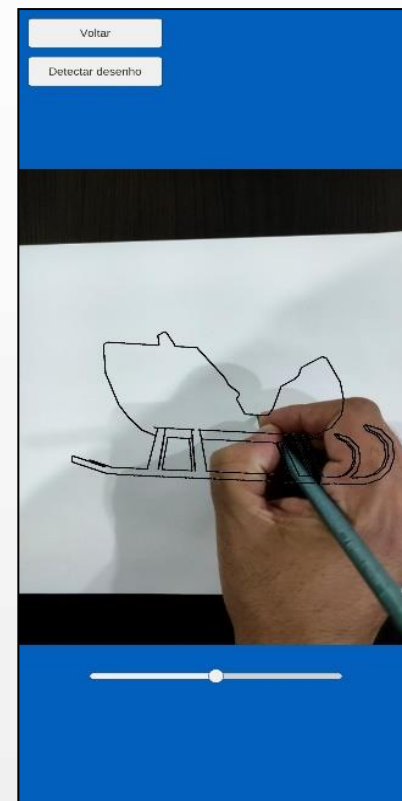
- Método GetTransparentTexture



1) Textura original



2) Textura após passar pelo método GetTransparentTexture



Análise dos Resultados

- Testes dos assets (OpenCVForUnity)
 - ArUco
 - Marker Less AR Example
- Teste de funcionalidade

Análise dos Resultados

- Teste de usabilidade
 - Dificuldade em importar arquivos em tempo de execução
 - Dificuldade em detectar desenho



1) Não detecção dos pontos



2) Detecção dos pontos

Conclusões

- Ferramenta Unity3D
- Biblioteca OpenCV integrado no asset OpenCVForUnity
- Biblioteca ArUco

Sugestões

- Estudo para melhorar a detecção do desenho
- Alterar as cores do objeto 3D, conforme o usuário pinte o desenho na folha de papel
- Melhorar o desempenho da câmera de RA sem marcadores predefinidos, quando tiver muitos objetos criados
- Importar outros formatos de arquivos 3D
- Adicionar mais de um objeto 3D na cena
- Melhorar a interface mostrada para o usuário

Demonstração

Aplicativo

Celular