Ferramenta de Realidade Aumentada utilizando Kinect para Estudos Topográficos

Bruno dos Santos Belaguarda[1], Alessandro André Mainardi de Oliveira[2], Gustavo Stangherlin Cantarelli[2]

[1]Acadêmico do Curso de Ciência da Computação – Universidade Franciscana (UFN)
Caixa Postal 151 – 97.010-032 – Santa Maria – RS – Brazil

[2]Docente dos Cursos de Ciência da Computação e Sistemas de Informação – Universidade Franciscana (UFN)

{bruno.belaguarda@gmail.com, alessandroandre@unifra.br, gus.cant@gmail.com}

Abstract. *With the advancement of teaching technologies, the present work presents the proposal of an augmented reality software applied to a sandbox for topographic studies, with the aid of Kinect and a projector. For the development of this software, the C # programming language will be used, along with XAML. With this, was chosen to the agile methodology, FDD to create the necessary documentation for the development of this tool.*

Resumo. *Com o avanço das tecnologias de ensino, o presente trabalho apresenta a proposta de um software de realidade aumentada projetada em uma caixa de areia para estudos topográficos, com auxílio do Kinect e um projetor. Para o desenvolvimento desse software, será usada a linguagem de programação C# junto a XAML. Com isso, foi escolhida à metodologia ágil, FDD para criar a documentação necessária para o desenvolvimento desta ferramenta.*

1. Introdução

O uso de novas tecnologias de ensino, em escolas, universidades e em outras instituições vem aumentando, mostrando-se um método mais prático e atrativo para alunos e professores, tornando a aprendizagem mais entendível para o aluno, fazendo com que o mesmo tenha um maior interesse pelo assunto tratado, e para o professor uma maior facilidade de ensino devido a familiaridade que os jovens tem com ambiente tecnológico, isso agregando mais conhecimento aos dois lados [G1 2018].

Conforme Di Maio (2011), existe um desafio de uso de novas tecnologias para professores no ensino da geografia, ou seja, da topografia consequentemente, com a familiarização dos jovens com a tecnologia este trabalho propõe além de apenas um ensino teórico dentro de uma sala de aula, um contato prático com o que foi aprendido teoricamente, com isso colaborando para uma maior absorção do conteúdo por parte dos alunos.

Assim foi desenvolvido um software, que se engaje nesse meio tecnológico de ensino, tendo como enfoque a aprendizagem em topografia, para isso foi usada a ideia de realidade aumentada com sobreposição de efeitos digitais que foram projetados em

cima de uma caixa com areia, oferecendo uma maneira prática e atrativa de explorar os processos da topografia. Além disso, foi utilizado para o desenvolvimento do mesmo, a linguagem de programação C#, em conjunto com XAML para criação da interface, utilizando à *User Interface* WPF (*Windows Presentation Fundation*), para seu desenvolvimento.

1.1. Objetivo Geral

Desenvolvido um software integrado ao Kinect com potencial para automatizar o processo de avaliações topográficas, possibilitando ao professor demonstrar na prática o aprendizado teórico, mostrando as técnicas da topografia e relevos, facilitando a interação professor aluno.

1.2. Objetivos específicos

Os objetivos específicos do presente trabalho são os seguintes:

- Estudo do funcionamento do Microsoft Kinect;
- Estudo do Kit de Desenvolvimento (SDK) da Microsoft;
- Estudo sobre a linguagem de programação C# e XAML;
- Estudo dos sensores do Microsoft Kinect;
- Estudo sobre o padrão de cores RGB;
- Estudo sobre topografia, altimetria, relevos;
- Desenvolver uma aplicação para auxilio no ensino da topografia.

2. Referencial Teórico

A seguir nesta seção serão apresentados conceitos relativos a área da geografia e tecnológica, que serão necessários para a compreensão desse trabalho.

2.1. Topografia

Segundo Veiga *et al.* (2007) origem da palavra topografia vem da palavra TOPOS, em grego, que significa lugar e GRAPHEN que significa descrição, assim, de uma forma bastante simples, Topografia significa descrição do lugar. Algumas definições:

> *"A Topografia tem por objetivo o estudo dos instrumentos e métodos utilizados para obter a representação gráfica de uma porção do terreno sobre uma superfície plana" [DOUBEK 1989].*

> *"A Topografia tem por finalidade determinar o contorno, dimensão e posição relativa de uma porção limitada da superfície terrestre, sem levar em conta a curvatura resultante da esfericidade terrestre" [ESPARTEL 1987].*

Portanto a topografia tem por finalidade determinar o contorno, dimensão e posição relativa de uma porção limitada de superfície terrestre, sem levar em conta a curvatura da terra. Ou seja, sempre será possível figurar em um plano horizontal a

imagem da região considerada representando todas as suas particularidades notáveis, naturais ou artificiais do terreno.

Assim conforme Espartel (1987), a topografia é uma ciência aplicada baseada na Geometria e na Trigonometria, onde a sua representação pode conter todos os detalhes da configuração do solo, mesmo que esse tenha detalhes artificiais como canais, estradas, prédios, etc. Além disso a topografia é subdividida em dois tipos de casos Planimetria e Altimetria.

A Planimetria é a representação em projeção horizontal dos detalhes existentes na superfície, ou seja, estuda a posição dos pontos e medidas, considerando todos em um mesmo plano [Espartel 1987].

A altimetria permite fixar, por meio de cotas ou quaisquer sinais convencionais, o relevo do terreno, isto é, a expressão exata de sua forma, ou seja, diferente da planimetria que estuda uma representação a partir de plano horizontal, a altimetria faz a determinação das alturas dos pontos topográficos em relação a um plano vertical de referência [Espartel 1987].

Na altimetria é usado um sistema representativo de cores baseados no padrão RGB, que serão representadas nas escalas do respectivo relevo onde acontece a mudança de cores como no exemplo da Figura 1.

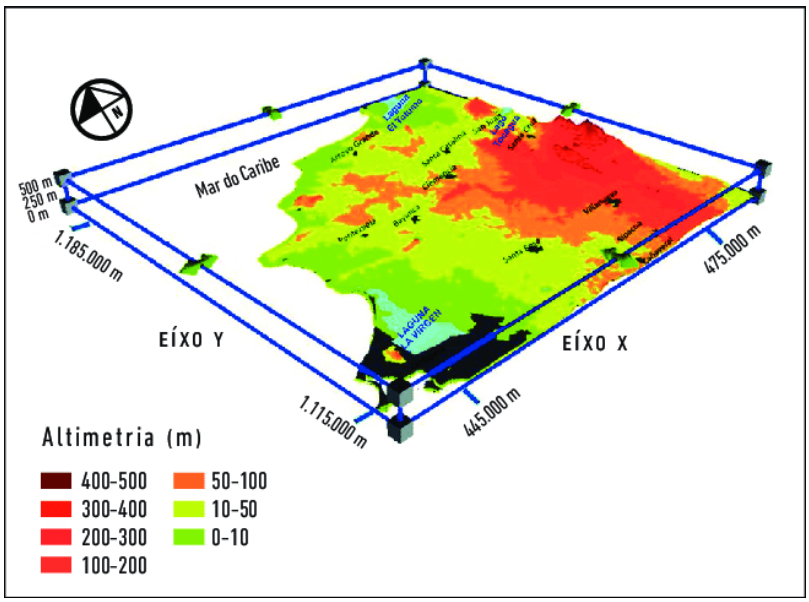

Figura 1. Imagem mostrando a representatividade de cores usado em um relevo [Furrier 2018].

2.2. Relevo

O relevo é onde as transformações geológicas se expressam mais nitidamente, conforme Bertolini *et al*. (2009), o relevo é um aspecto da natureza e faz parte do espaço físico que exerce grande fascínio sobre os olhares atentos à paisagem, suas dinâmicas de formação tem base em suas fisionomias externas que são montanhas, planaltos, planícies e depressões.

2.3. Microsoft Kinect

O dispositivo Kinect (Figura 2), foi desenvolvido pela Microsoft para games, onde substituiria o apertar de botões de um controle, pelos movimentos do jogador. Lançado em novembro de 2010 como um acessório para o console Xbox360 e para competir com o WII da Nintendo [Canaltech 2015].

Figura 2. Imagem mostrando detalhes do Microsoft Kinect [Canaltech 2015].

O Kinect possui um sofisticado algoritmo de processamento paralelo necessário para extrair o mapa de profundidade a partir da luz estruturada recebida. E segundo Crawford (2010), para possuir mais precisão nas informações dos sensores, as imagens são alinhadas pixel a pixel, ou seja, cada pixel de imagem colorida é alinhado a um pixel da imagem de profundidade. Além disso, o Kinect sincroniza (em tempo real) todas as informações dos sensores (profundidade, cores e áudio) e as entrega através do protocolo Serial. A tecnologia inovadora por trás do Kinect é uma combinação de software e hardware. Além disso Crawford (2010) cita três elementos inovadores no hardware do sensor Kinect:

- **Câmera de vídeo VGA colorida** - Esta câmera de vídeo ajuda no reconhecimento facial e em outros recursos de detecção, detectando três principais cores: vermelho, verde e azul. A Microsoft chama isso de "câmera RGB", referindo-se aos componentes de cor que ele detecta.
- **Sensor de profundidade** - Um projetor infravermelho e um sensor monocromático, trabalham juntos para se ajustar ao ambiente e “enxerga-lo” em 3-D, independentemente das condições de iluminação.
- **Microfone multi-matriz** - Esta é uma série de quatro microfones que podem isolar as vozes dos jogadores do ruído no ambiente. Isso permite que a uma distância considerável do Kinect o usuário ainda possa usar controles de voz.

O principal item do sensor Kinect que será utilizado para o projeto, será seu sensor de profundidade (Infravermelho), este permite o Kinect escanear o ambiente, retornando dados de distâncias. Estes dados serão usados como modelo de altitudes, para serem trabalhados junto com as cores do sistema RGB.

Além disso foi utilizado o Kit de Desenvolvimento de Software (SDK) do Kinect para Windows permite que desenvolvedores criem aplicativos que reconheçam gestos, voz, usando a tecnologia dos sensores do Kinect [MICROSOFT 2003].

2.4. Projetor multimídia

O projetor multimídia é um dispositivo óptico mecânico que é capaz de produzir imagens a partir de TVs, computadores, notebook, etc. Através de uma tecnologia denominada DLP (*Dual Light Processing*) em alguns, e outros com a tecnologia LCD (*Liquid Cristal Display*), estes equipamentos tem uma lente convergente (objetiva), que

fornece imagens reais, invertidas e maiores que o objeto, podendo ser de um slide ou filme [Alecrim 2007].

2.5. Microsoft Visual Studio

Microsoft Visual Studio é um ambiente de desenvolvimento integrado da Microsoft para desenvolvimento de software utiliza-se das linguagens C, C++, C# (C Sharp) e J# (J Sharp) [Microsoft 2018].

2.5.1. WPF

O WPF ou (*Windows Presentation Foundation*), é um sistema subgráfico do .NET *FRAMEWORK*, criado inicialmente para criação de programas direcionados ao *Windows Vista*.

Ainda segundo Moreira (2009), de uma forma geral um programa criado em WPF é composto por duas partes, uma parte criada em XML, chamada de XAML, onde fica a aparência do programa desenvolvido, ou seja, à interface gráfica, e as funcionalidades dessas interfaces, implementadas em C#. O WPF suporta interação com interfaces de aplicação em 2D e 3D.

2.5.1.1. C Sharp

C Sharp ou (C#) é uma linguagem de programação orientada a objetos desenvolvida pela Microsoft, faz parte da plataforma .NET e está entre as mais utilizadas no mundo. Apesar de possuir uma sintaxe bem parecida com outras linguagens populares como C, C++, Java e *Object Pascal*, foi uma linguagem criada do zero, mesmo assim possui muitos elementos da linguagem pascal e Java. De acordo com a Microsoft (2016), C# é compilado para *Common Intermediate Language* (CIL) que é interpretado pela máquina *virtual Common Language Runtime* (CLR). É uma linguagem de programação multi-paradigma fortemente tipada.

2.5.1.2. XAML

Conforme Marcoratti (2015), XAML é uma linguagem de marcação declarativa, ou seja, ela simplifica a criação de uma interface de usuário para um aplicativo feito em WPF. O XAML além de ser baseado em XML, representa diretamente a instanciação de objetos em um conjunto específico de tipos de suporte definidos em *assemblies*. Isso é diferente da maioria das outras linguagens de marcação, que são geralmente uma linguagem interpretada sem um vínculo direto para um sistema de tipos de suporte. Ou seja, XAML habilita um fluxo de trabalho em partes separadas que podem trabalhar juntas na interface do usuário e na lógica da aplicação, usando ferramentas potencialmente diferentes no caso C# e XML juntos na mesma aplicação.

2.6. FDD

Feature Driven Development – FDD, foi criado em 1997, por Peter Coad a partir de experiência de análise e modelagem. E segundo Goyal (2007) , o FDD não possui uma restrição quanto ao tamanho da equipe, que neste projeto foi executado por somente uma pessoa. Além disso Goyal (2007) descreve o FDD em cinco processos como mostra a Figura 3.

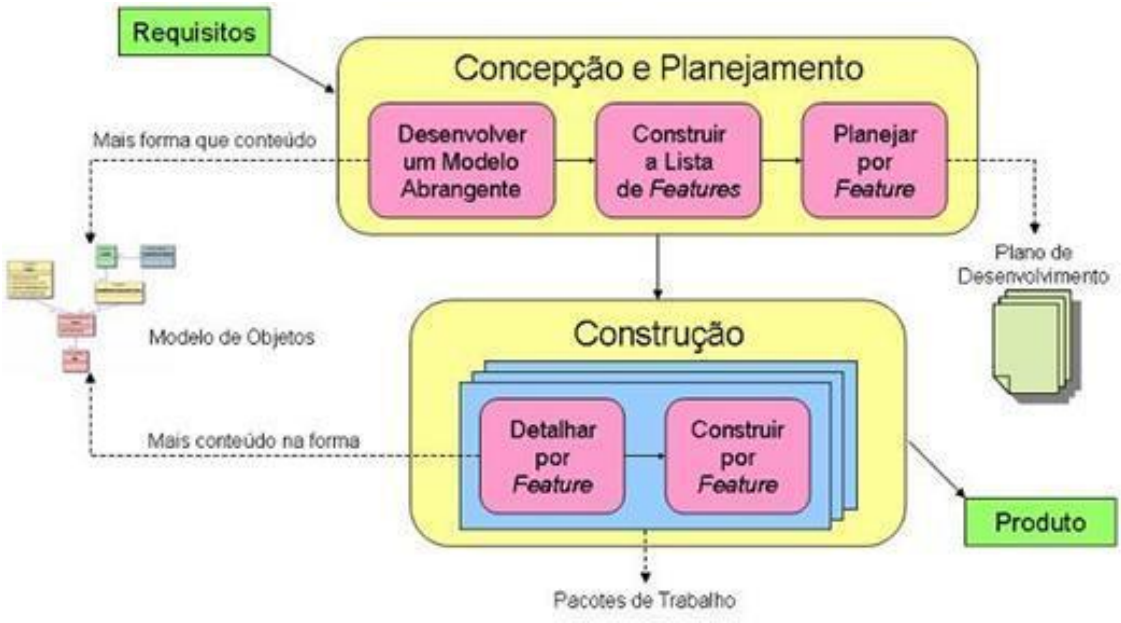

Figura 3. Processos do FDD [Heptagon 2010].

Como foi mostrado na Figura 3 as etapas do FDD, onde a primeira é de concepção e planejamento, executada apenas uma vez durante todo o projeto, e a fase de construção que ocorre enquanto existirem funcionalidades a serem implementadas. Essas duas etapas são formadas por cinco processos, que serão descritos abaixo, e são mostrados na Figura 3.

- Desenvolver um modelo abrangente: Aqui se faz o entendimento do produto através dos requisitos, junto as especificações de cada funcionalidade.
- Construir a lista de funcionalidades: Fase em que é criada uma lista de funcionalidades, através do que foi identificado na etapa anterior, devendo cada item dessa lista ter a aceitação do cliente.
- Planejar por funcionalidades: Aqui cada função criada deve ser executada, seguindo suas prioridades.
- Detalhar por funcionalidades: Onde cada funcionalidade vai ser modelada de acordo com as necessidades, como: protótipos de telas, diagramas de domínio, etc.
- Construindo por funcionalidades: Fase final onde o Código de cada funcionalidade do produto deve ser desenvolvido.

A partir disso e obedecendo aos processos básicos estabelecidos pela metodologia FDD, a primeira etapa para o desenvolvimento do projeto sendo a de concepção e planejamento, e a segunda etapa sendo a de construção.

3. Trabalhos correlatos

Nesta seção, são descritos trabalhos similares a proposta do presente trabalho.

3.1. SandBox

Segundo Kreylos (2016), o projeto *SandBox* combina aplicativos de visualização 3D com uma exposição prática para ensinar conceitos geográficos. A caixa de areia de realidade aumentada (AR) permite que os usuários criem modelos de topografia moldando a areia real, que é então aumentada em tempo real por um mapa de cores de elevação, linhas de contorno topográficas e água simulada. O sistema ensina conceitos geográficos, geológicos e hidrológicos, como a leitura de um mapa topográfico, o significado de linhas de contorno, bacias hidrográficas, áreas de captação, diques, etc.

A aplicação SandBox funciona apenas no Sistema Operacional Linux Mint, utiliza de um kit de ferramentas para desenvolvimento das cores chamado Vrui Vr Toolkit, o kit de ferramenta do Kinect, um projetor e computador, como mostra a Figura 4.

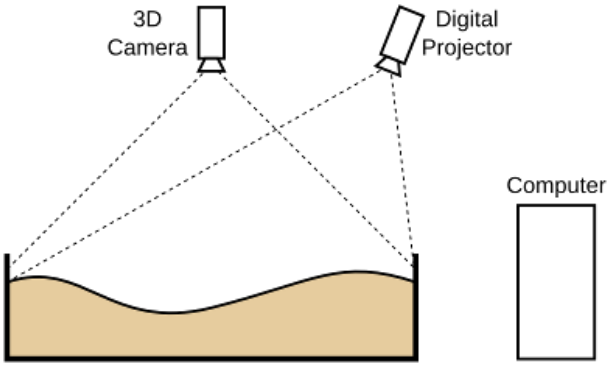

Figura 4. Layout simplificado da sandbox [Kreylos 2016].

A Figura 5 mostra o exemplo do projeto *SandBox* já concluído.

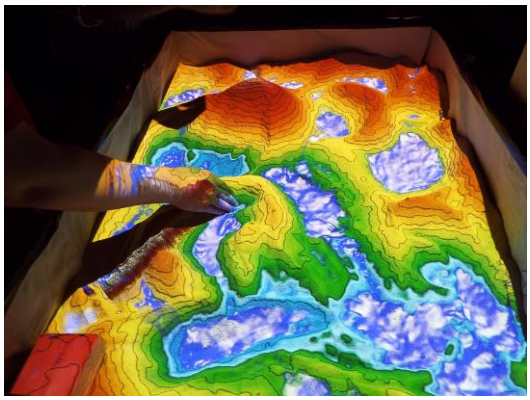

Figura 5. Sandbox em funcionamento. [Kreylos 2016].

A sua diferença com a proposta desse trabalho, é que em vez de ser uma aplicação que funcione apenas no *Linux mint* como o *SandBox* e use ferramentas para criação das cores, a aplicação proposta no trabalho foi desenvolvida usando C# e XAML, então a intenção que com desenvolvimento deste software seja facilitado o uso

aos usuário e também com seu funcionamento em *Windows*, seja alcançando uma maior quantidade de usuários já que conforme o site G1 (2017) o *Windows* ainda é o sistema operacional mais usado em 84% dos computadores.

4. Metodologia

Esta seção tem por objetivo seguir cada processo do desenvolvimento do projeto, seguindo os padrões da metodologia FDD.

Segundo o autor Sommerville (2011), os métodos ágeis "destinam-se a entregar um software rapidamente aos clientes", já que legislação muda frequentemente, e a metodologia ágil em geral conta com interação para certas especificações, assim sendo capaz de acompanhar o desenvolvimento no qual os requisitos tem mudanças tão frequentes. Com isso é de fundamental importância que se tenha um planejamento e uma organização, para auxiliar no desenvolvimento do projeto. E a fim de ter uma boa organização, documentação e otimização do tempo, é necessário que se utilize uma metodologia adequada.

4.1 Projeto do software

Para o desenvolvimento do software foi usado a UI (*User Interface*) WPF, e definiu-se C# como linguagem de programação e XAML para desenvolvimento da interface, já a metodologia ágil que foi utilizada, a *Feature Driven Development* (FDD), com intuito de obter um melhor resultado, pois esta metodologia atende as necessidades demandadas deste trabalho.

4.1.1. Desenvolvendo um modelo abrangente

Responsável por apresentar uma visão geral do objetivo do projeto, é observado o Diagrama de Domínio (Apêndice A).

4.1.2. Requisitos Funcionais e Não Funcionais

Aqui são apresentados os requisitos funcionais e não funcionais do sistema, assim como uma descrição e a complexidade dos mesmos. Seguindo a metodologia utilizada, a seguir a tabela de requisitos funcionais Tabela 1 e Requisitos Não Funcionais Tabela 2, presentes no sistema:

Tabela 1. Requisitos Funcionais

Requisitos Funcionais (RF)		
Funcionalidade	Descrição	Complexidade
RF 1: Gerenciar cores	O sistema deve manipular o sistema de cores RGB, em conjunto a altimetria.	Alta
RF 2: Gerenciar conexão com Kinect	O sistema deve reconhecer o Kinect.	Média
RF 3: Gerenciar Ajuste do Kinect	O Sistema deve fornecer ajuda para usuário ajustar o ângulo do Kinect	Média

Tabela 2. Requisitos Não Funcionais

Requisitos Não Funcionais (RNF)	
Funcionalidade	Descrição
RNF 1:	O sistema deve ser desenvolvido para o sistema operacional *Windows*.
RNF 2:	O sistema deverá ser desenvolvido na linguagem de programação C#.
RNF 3:	O sistema deverá ter um menu.
RNF 4:	O Sistema deverá ter uma opção chamada "Ajuda", para informar dicas de uso ao usuário.

4.1.3. Planejamento por funcionalidade

Na etapa de planejamento por funcionalidade foi definido o ordenamento para implementação, onde será estimado um tempo de desenvolvimento para cada funcionalidade da ferramenta (Apêndice B).

4.1.4. Detalhando por funcionalidade

Nesta etapa, foi desenvolvido o Diagrama de Caso de Uso, com o objetivo de obter uma melhor visão do cenário da aplicação, sendo possível visualizar as ações do ponto de vista do usuário (Apêndice C).

Segundo Booch (2006) o Diagrama de Atividade, é um diagrama que demonstra o fluxo, ou seja, mostra o fluxo de controle de uma atividade para outra, e a interação entre o usuário, o sistema, o Kinect e o projetor (Apêndice D).

4.1.5. Construindo por funcionalidade

Na última fase da metodologia será mostrado Diagrama de Classes que ilustra um conjunto de classes, interfaces, colaborações e as relações entre os objetos. Para mostra essa estrutura a Figura 6.

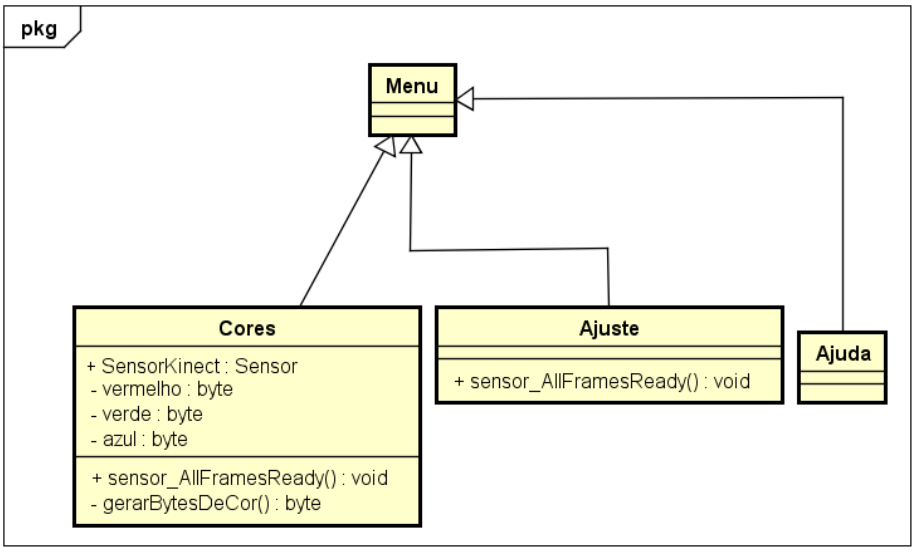

Figura 6. Diagrama de Classe.

5. Implementação

A implementação do software foi realizada utilizando o WPF(*Windows Presentation Foundation)*, que separa o sistema em partes que facilitam o seu desenvolvimento, separado na programação em C# e as interfaces usando XAML.

A partir disso, a principal função do software começa com o processamento de cores, o que dará a ideia da altimetria, para isso é usado um Kinect, este que retorna os dados de distância que são capturados pela sua câmera, através de um método chamado *DepthImageFrame*, este método informa acesso as dimensões da imagem, os dados e permite o mapeamento das coordenadas, e retorna valores de distância e detecção de jogadores, no trecho de código mostrado na Figura 7, pode ser visto a parte de verificação do *DepthImageFrame.* onde a partir de que ele exista, seja gerado bytes coloridos para imagem, após isso é recebido os valores de distância e os valores das cores, então estes valores são passados para a imagem.

```
void sensor_AllFramesReady(object sender, AllFramesReadyEventArgs e)
{
    // o depthFrame e pegado
    using (DepthImageFrame dframe = e.OpenDepthImageFrame())
    {
        // Caso não receba nenhum frame sai do método
        if (dframe == null)
            return;

        // Parte onde gera os bytes coloridos  para a nossa imagem
        byte[] pixels = GerarBytesDeCor(dframe);

        // Numero de bytes por linha largura * 4 (R,G,B, Vazio)
        int stride = dframe.Width * 4;

        // Cria a imagem passando o tamanho da imagem  o dpi e o formato (32bits)
        // a paleta (nula) o array de bytes da imagem (bytesp) e o stride


        imgDepth.Source = BitmapImage.Create(640, 480, 96, 96, PixelFormats.Bgr32, null, pixels, stride);


    }
}
```

Figura 7. Código de conexão com a câmera de profundidade do Kinect.

No próximo trecho de código Figura 8, é mostrado como é conseguido as distâncias de profundidade, através da variável *depth* no método *GerarBytesDeCor*, este *depth* recebe apenas os bits que definem os valores referentes a distância, ou seja, a profundidade, com esses dados em milímetros. Após isso são definidas as variáveis onde serão informados os valores das cores: vermelho, verde e azul. Então é definido que a cada 1cm de distância será informada uma cor, tirando a base e o topo que são distâncias diferentes das demais. Os valores das cores informadas a cada 1cm de distância são de cores usadas em mapas altimétricos.

```
private byte[] GerarBytesDeCor(DepthImageFrame dFrame)
{
    // tamanho do frame do Depth
    // cada ponto tem 16bits
    short[] rawDepthData = new short[dFrame.PixelDataLength];

    // Copia a informação para o raw
    dFrame.CopyPixelDataTo(rawDepthData);

    //o array de byte sera a imagem

    Byte[] pixels = new byte[dFrame.Height * dFrame.Width * 4];

    // Posição das cores
    int colorPixelIndex = 0;

    //  profundidade
    short depth = 0;


    // Variaveis onde é jogado o valor das cores
    byte azul, vermelho, verde;

    // Um loop para converter cada ponto de profundidade em cor
    for (int i = 0; i < rawDepthData.Length; ++i)
    {
        // Pega só a parte que contem a informação da profundidade, descartando os outros dados
        depth = (short)(rawDepthData[i] >> DepthImageFrame.PlayerIndexBitmaskWidth);

        // zera as variaveis
        azul = 0;
        vermelho = 0;
        verde = 0;



        // 90 CM
        /*Cores definidas através da distancia informada a cada 2cm */
        if (depth > 900 && depth < 940)
        {

            azul = 162;
            verde = 68;
            vermelho = 45;
        }
        else if (depth > 890 && depth < 900)
        {
            azul = 232;
            verde = 162;
            vermelho = 0;
        }

        else if (depth > 880 && depth < 890)
        {
            azul = 151;
            verde = 117;
            vermelho = 47;
        }
        else if (depth > 870 && depth < 880)
        {
            azul = 0;
            verde = 255;
            vermelho = 156;
        }
```

Figura 8. Trecho de código, onde é feito a saída das cores.

Após as implementações feitas, foi desenvolvida as interfaces, a primeira um menu, contendo três opções: Iniciar, Câmera e Ajuda como pode ser visto na Figura 9.

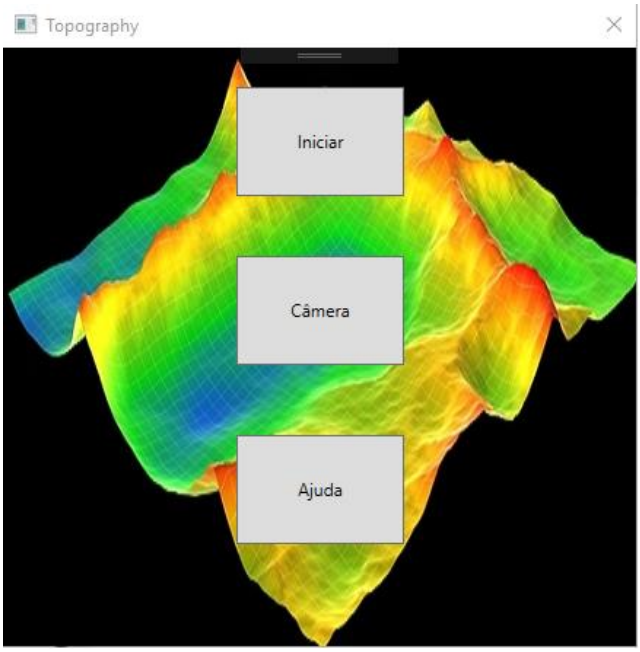

Figura 9. Interface do menu.

Com isso selecionando a opção “Iniciar”, é começado o trabalho com as cores onde estas serão projetadas em cima da areia, o que dará a ideia de realidade aumentada. A partir disso movimentando a areia, poderá ser criado diferentes tipos relevos, que serão atualizados em “tempo real” a cada movimento, podendo também ser vista as atualizações na tela do computador. Então ao selecionar a opção “Câmera”, será mostrado a imagem real sem os dados de profundidade, isso para melhor visualização e ajuste do mesmo como mostra a Figura 10, um comparativo com o “Iniciar” e abaixo a “Câmera”.

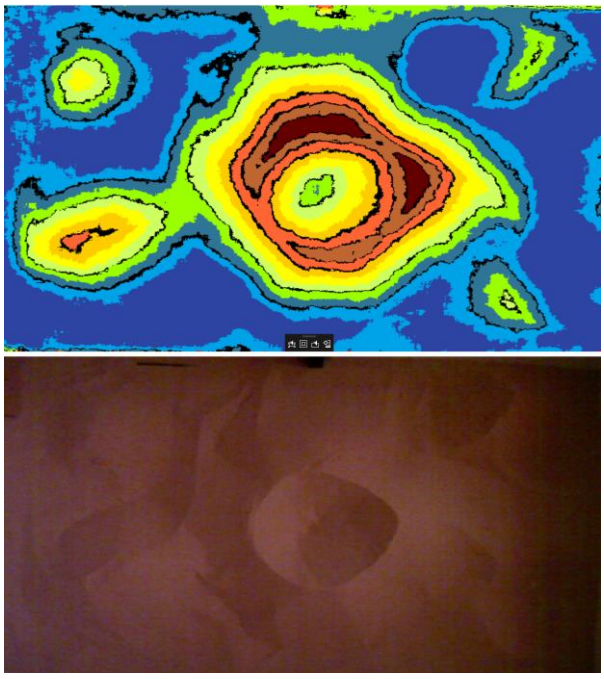

Figura 10. Comparativo das interfaces da “Câmera” e “Iniciar”, ambas em *“fullscreen”*.

Ao selecionar a opção “Ajuda”, será mostrada uma janela com dicas, como distância ideal do Kinect, ajustes no projetor, tamanho de caixa entre outras dicas, também é mostrado um mapa retirado do projeto em execução, com os níveis altimétricos se fossem em uma escala real, e as cores utilizadas no projeto, Figura 11.

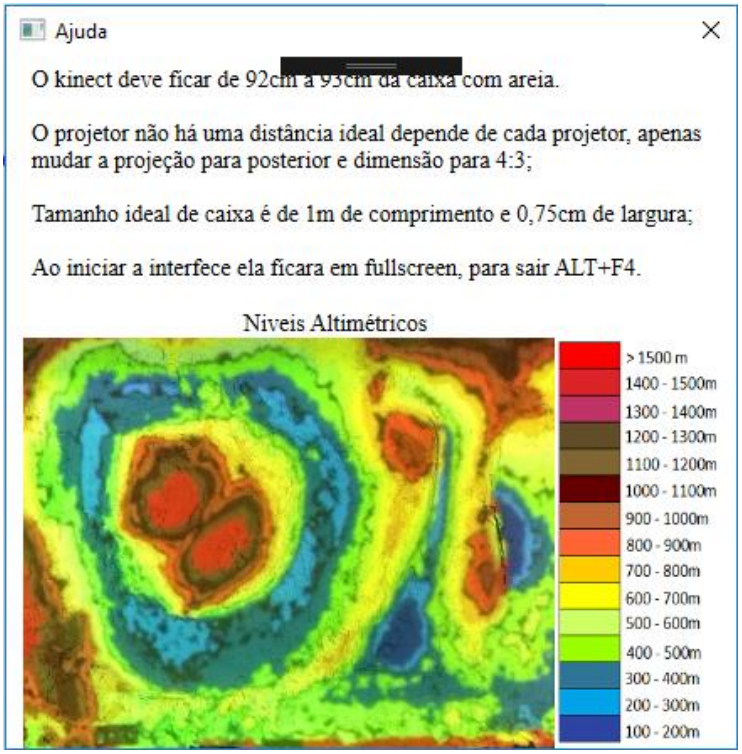

Figura 11. Interface de ajuda.

6. Resultados

Após a implementação, deu-se início ao desenvolvimento do projeto, onde foi construído uma caixa com 1m de comprimento e 0,75cm de largura, deixando o Kinect a 91cm da base da caixa, e o projetor a 1,30m.

Os resultados obtidos foram os desejados, onde com a projeção feita pelo projetor teve-se a sensação de realidade aumentada e em “tempo real”, podendo ser modificadas as formas da areia para criação de relevos como planícies, planaltos, montanhas entre outros, isso com a ideia de altimetria inclusa, onde são mostrado os níveis de altitude em cores no relevo criado na areia, como pode ser visto na Figura 12.

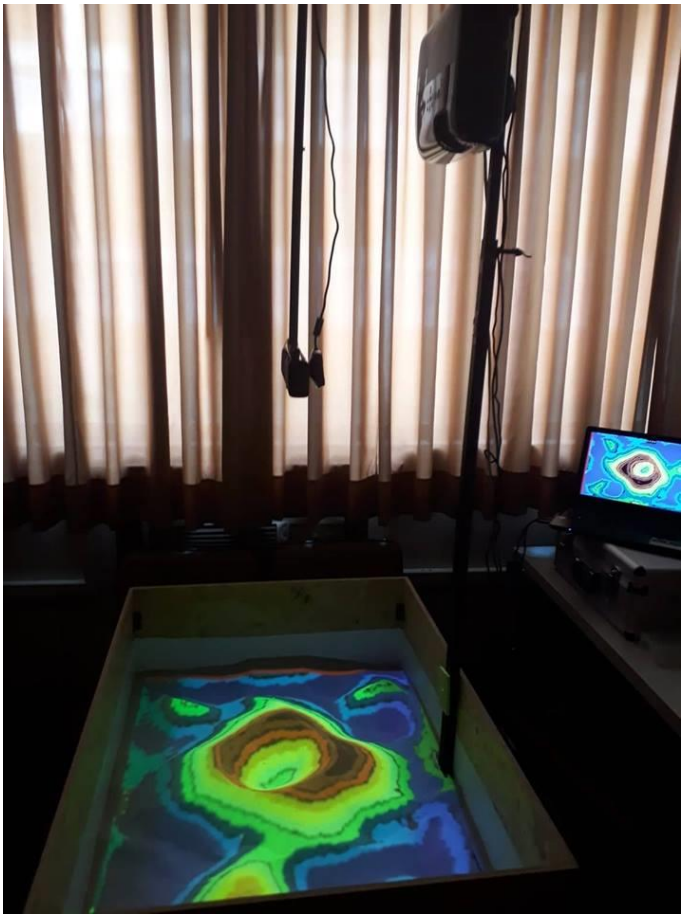

Figura 12. Projeto desenvolvido.

7. Conclusão

Com o desenvolvimento da proposta deste trabalho, tem se uma nova ferramenta tecnológica para o ensino na área da Geografia em instituições de ensino, e que o desenvolvimento feito para Windows facilite o seu uso, abrangendo uma área maior. Busca-se, também que quando os usuários remodelem a areia, eles acabem reconhecendo suas ações e acabem sendo atraídos pela ferramenta que dará a percepção de uma realidade aumentada, e acabe servindo para demonstrar uma ampla gama de conceitos sobre a geografia, isso interativamente.

Para isso foi usada uma sobreposição de efeitos digitais que serão projetados em cima de uma caixa com areia, oferecendo uma maneira prática e atrativa de explorar os processos da topografia. Além disso, será utilizado para o desenvolvimento do mesmo, a linguagem de programação C#, em conjunto com XAML para criação da interface, utilizando à *User Interface* WPF, para seu desenvolvimento.

Subsequente a isso, haveria a possibilidade de implementar uma ferramenta de dados planimétricos, como distâncias entre pontos, isso a partir de uma implementação de escalas, também poderiam ser adicionados efeitos de água, e chuva como são mostrados no exemplo do trabalho correlato.

Referências

Alecrim, Emerson (2007) “Projetores de vídeo: principais características”. Disponível em < https://www.infowester.com/projetores.php>. Acesso em 7 de maio 2018.

Bertolini et al (2009), “A abordagem do relevo pela Geografia: uma análise a partir dos livros didáticos”, Disponível em < http://ppegeo.igc.usp.br/index.php/TED/article/view/8364/7635 > Acessado em 26 de Abril 2018

Booch G. (2006) UML – Guia do Usuário, Elsevier, 1ª edição.

Canaltech, (2015) “Como Funciona o Kinect”. Disponível em <https://canaltech.com.br/games/Como-funciona-o-Kinect/>.Acesso em 10 de abril 2018.

Crawford, Stephanie. “How Microsoft Kinect Works”. Disponível em: < https://electronics.howstuffworks.com/microsoft-kinect2.htm>. Acesso em 10 de abril 2018.

Di Maio, A. C., “Educação, Geografia e o desafio de novas tecnologias”, Revista Portuguesa de Educação, (2011), 24(2), pp. 211-241.

Espartel, L. Curso de Topografia. 9 ed. Rio de Janeiro, Globo, (1987).

Furrier, Max (2018) ”Geomorfologia estrutural, morfotectônica e morfometria da folha Cartagena 1:100.00-Colombia”, Disponível em < https://www.researchgate.net/publication/323464766_Geomorfologia_estrutural_morfotectonica_e_morfometria_da_folha_Cartagena_1100000-Colombia?_sg=DJKAugWlF_WBCnCzhWJOHWeVZhFghuMx5GVX0yVlqdhL3pisWK9THMAD1qNG-3o205LifHWzMQ > Acessado em: 15 maio 2018.

G1, (2017) “Android passa Windows e se torna o sistema operacional mais usado do mundo”. Disponível em < https://g1.globo.com/tecnologia/noticia/android-passa-windows-e-se-torna-o-sistema-operacional-mais-usado-do-mundo.ghtml >.Acesso em 7 de maio 2018.

G1, (2018) “Escolas usam tecnologia digital como ferramenta de ensino”. Disponível em <http://g1.globo.com/como-sera/noticia/2018/04/escolas-usam-tecnologia-digital-como-ferramenta-de-ensino.html>.Acesso em 9 de maio 2018.

Goyal, S. (2007) “Agile Techniques for Project Management and Software Engineering” Disponível em: <http://csis.pace.edu/~marchese/CS616/Agile/FDD/fdd.pdf> Acessado em: 09 maio 2018

Heptagon. (2010) “Feature Driven Development”, Disponível em < http://www.heptagon.com.br/fdd> , Acessado em: 09 maio 2018.

Kreylos, Oliver, 2016 “Augmented Reality Sandbox”. Disponível em < http://idav.ucdavis.edu/~okreylos/ResDev/SARndbox/index.html >.Acesso em 7 de maio 2018.

Marcoratti, José C., (2015) “.NET - Introdução a XAML(Extensible Application Markup Language)”. Disponível em < http://www.macoratti.net/09/03/net_xaml.htm >.Acesso em 7 de maio 2018.

Microsoft (2013) “Kinect para Windows ”. Disponível em < https://developer.microsoft.com/pt-br/windows/kinect>.Acesso em 7 de maio 2018.

Microsoft (2016) “Um tour pela Linguagem C#.”. Disponível em < https://docs.microsoft.com/pt-br/dotnet/csharp/tour-of-csharp/ >.Acesso em 7 de maio 2018.

Microsoft (2018) “ IDE Visual Studo.”. Disponível em < https://visualstudio.microsoft.com/pt-br/vs/>.Acesso em 7 de maio 2018.

Moreira, Helio, (2009) “Tutorias C# - O que é o WPF (Windows Presentation Foundation)”. Disponível em < https://pplware.sapo.pt/tutoriais/tutorial-c-o-que-e-o-wpf-windows-presentation-foundation/ >.Acesso em 7 de maio 2018.

Pressman, Roger S. Software Engineering: a Practitioner's Approach. 6. ed. McGraw-Hill, 2005

Rocha, João C., (2010) “Cor luz, cor pigmento e os sistemas RGB e CMY ”. Disponível em: <http://www.belasartes.br/revistabelasartes/downloads/artigos/3/cor-luz-cor-pigmento-e-os-sistemas-rgb-e-cmy.pdf>. Acesso em 7 de maio 2018.

Sommerville, I. (2011). Engenharia de *Software*. 9ª ed. São Paulo: SP. Addisson Wesley.

Veiga et al (2007), “Fundamentos de topografia”, Disponivel em <http://engenhariaconcursos.com.br/arquivos/Topografia/fundamentos%20de%20topografia.pdf> Acessado em 26 de Abril 201

Apêndice A - Desenvolvendo um modelo abrangente

Diagrama de Domínio poder conferido na Figura 13.

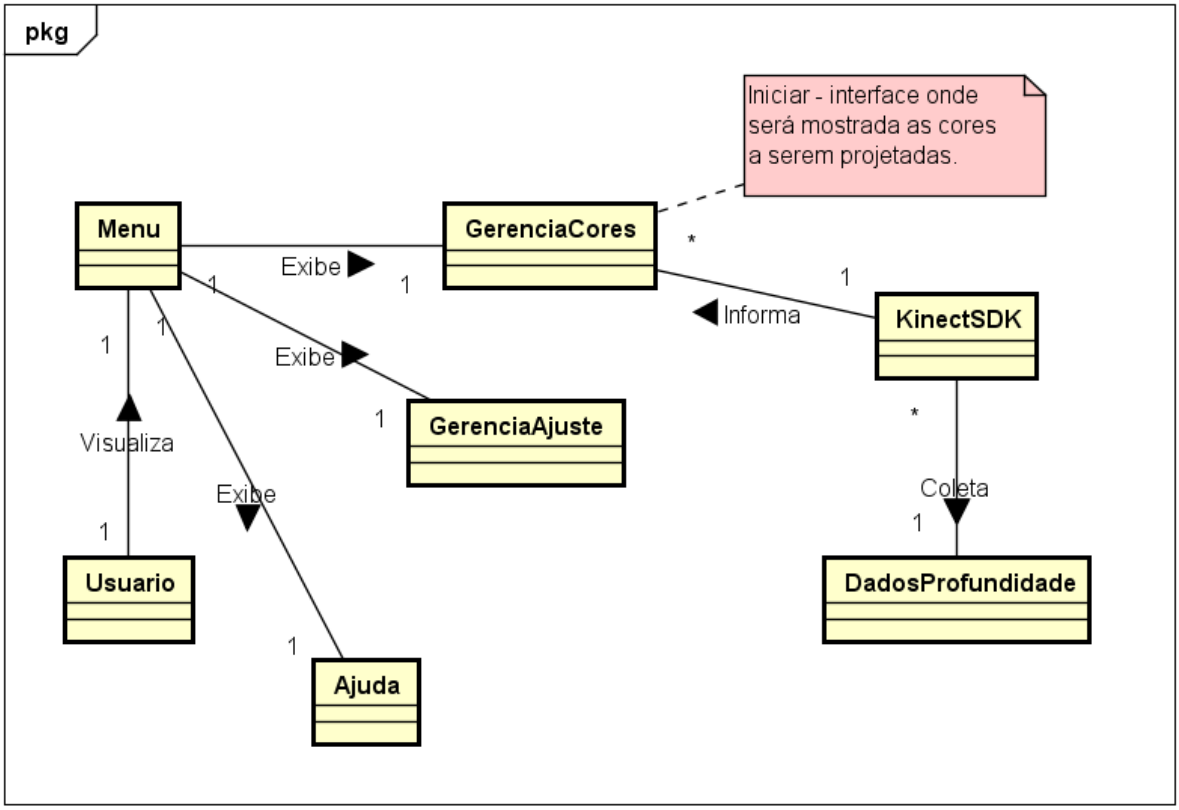

Figura 13. Diagrama de Domínio.

Apêndice B – Planejamento por funcionalidade

Tabela 3. Planejamento por Funcionalidade

Requisitos Funcionais (RF)	
Funcionalidade	Tempo
RF 1: Exibir cores	60 dias
RF 2: Gerenciar conexão com Kinect	1 dias
RF 3: Gerenciar Ajuste do Kinect	4 dias

A demanda de tempo elevada no RF1, deve-se por ser o principal requisito do projeto onde será manipulado o padrão RGB, aplicado altimetria e onde será manipulado os dados enviados pelo Kinect.

Apêndice C – Detalhando por funcionalidade (Caso de Uso)

O diagrama de Caso de Uso pode ser conferido na Figura 14.

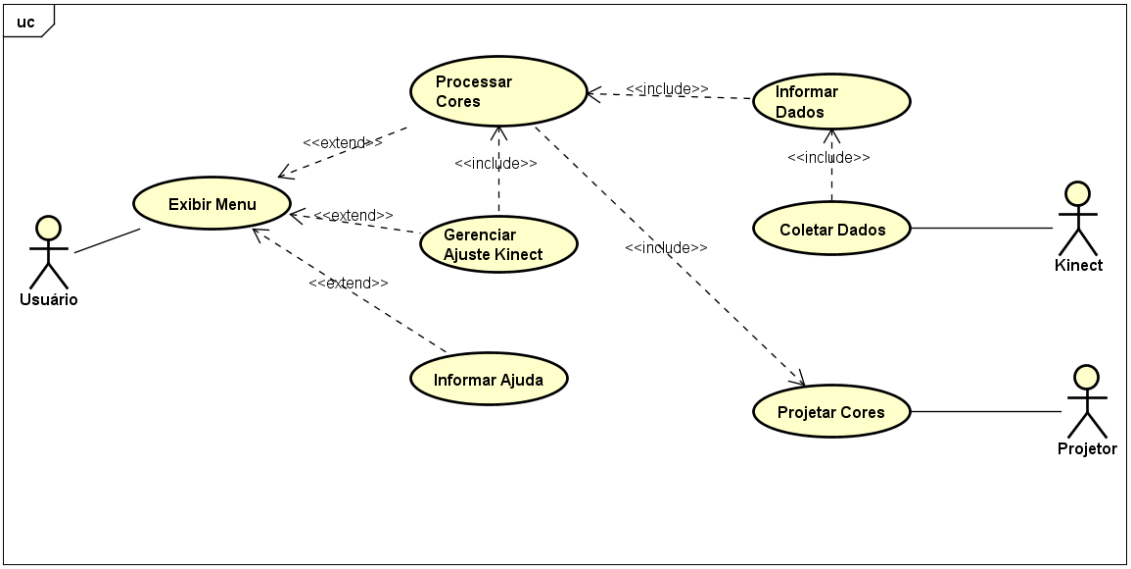

Figura 14. Diagrama de Caso de Uso.

Apêndice D – Detalhando por funcionalidade (Diagrama de Atividades)

O diagrama de Diagrama de Atividades pode ser conferido na Figura 15.

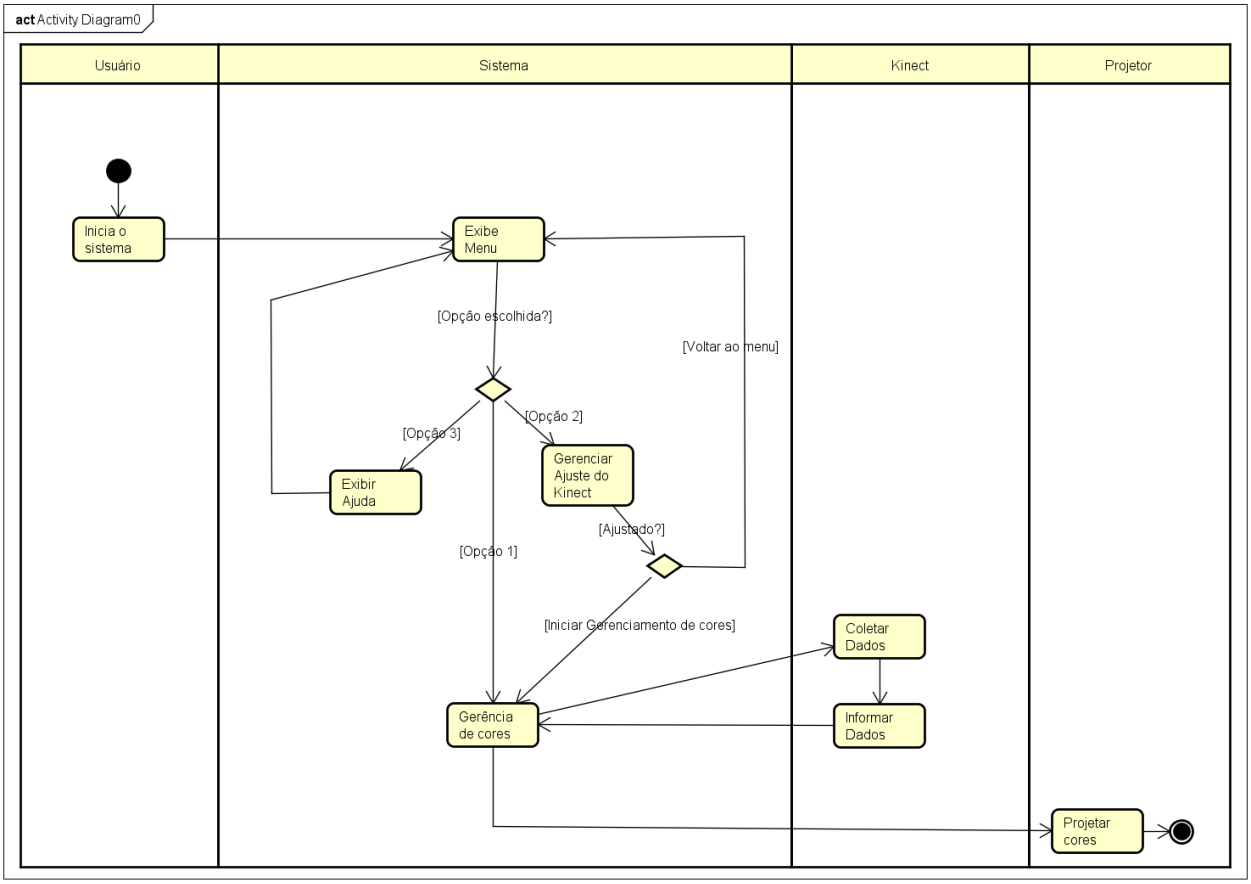

Figura 15. Diagrama de Atividades.

Apêndice E – Protótipos de interface

A seguir serão mostrados os protótipos de tela, Pressman (2005) afirma que mesmo havendo várias formas e metodologias para se iniciar a desenvolver um software, independente da escolha, se faz de forma crucial a etapa de prototipação para que haja sucesso no desenvolvimento do projeto, fazendo com que o desenvolvedor otimize seu tempo e reduza esforço, e também facilitando a visualização do produto para o cliente desde a fase inicial.

A seguir serão apresentados os esboços das interfaces a serem desenvolvidas para o *software* deste projeto. Antes de cada esboço uma breve descrição de sua funcionalidade.

O esboço na Figura 16, mostra uma interface menu, para facilitar a escolha que o usuário irá ter poder escolher entre as opções "Iniciar" para iniciar o gerenciamento de cores do *software*, "Ajustar" para ajustar o Kinect em relação a caixa de areia e "Ajuda" onde terá informações e dicas para o ajuste.

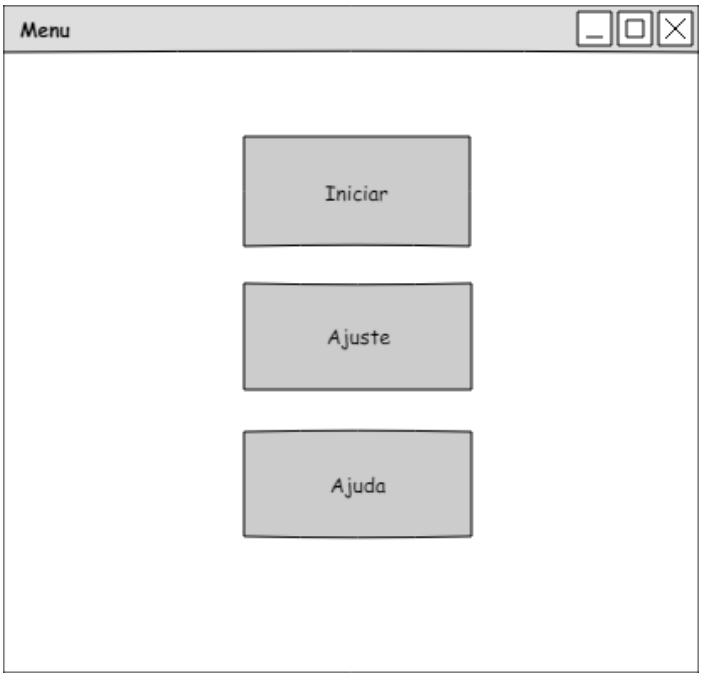

Figura 16. Protótipo da interface menu.

A seguir na Figura 17, a interface principal do sistema, onde está mostrará as cores que serão desenvolvidas, com o sistema RGB de cores, e a partir dos resultados de profundidade coletados pelo Kinect, assim será modificado a colorização que será projetada na areia.

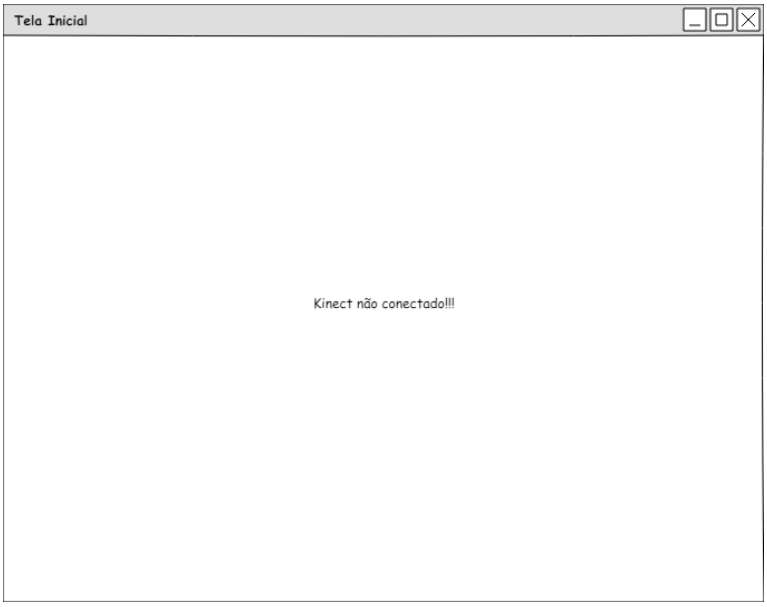

Figura 17. Protótipo da interface de processamento de cores, onde irá mostrar as cores do relevo.

A Figura 18, mostra a interface de ajuste do Kinect em relação a caixa, a partir da câmera do Kinect, o usuário poderá obter a distância do mesmo em relação a caixa, também o ângulo podendo ser mudado através de um componente de escala.

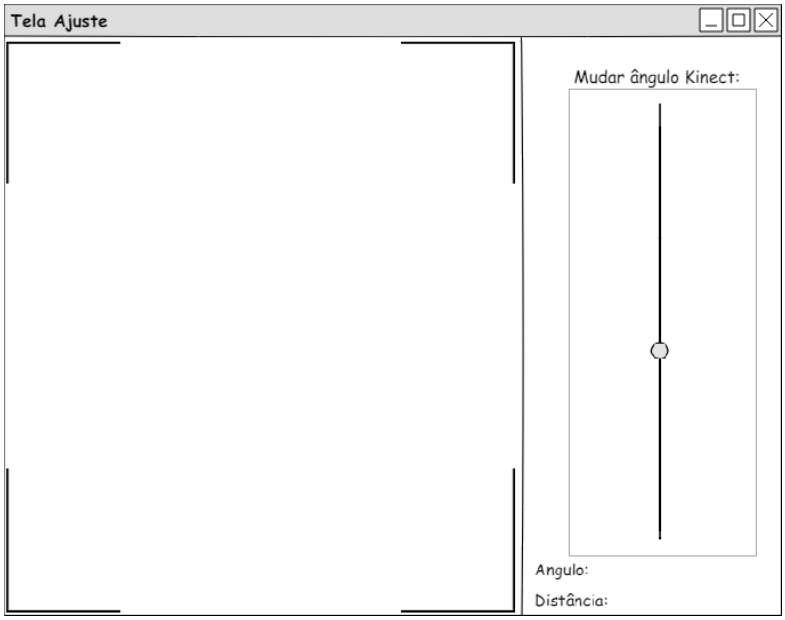

Figura 18. Protótipo da interface de ajuste.

A interface de ajuda irá conter apenas dicas para usuário e informações sobre o software, não foi necessário um protótipo para tal.

Apêndice F – Projeto

Portanto a Figura 18 irá mostrar como deve ficar o projeto após seu desenvolvimento, a ideia então é que o Kinect fique a certa distância da caixa, a partir disso com a câmera de profundidade ele irá coletar os valores de distância, que serão trabalhados com

conceitos RGB. Para aplicação a interface de processamento de cores será em *fullscreen,* o que irá possibilitar uma visualização melhor do usuário assim que projetada em cima da areia.

Figura 19. Esboço do projeto.