

INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA
DE MINAS GERAIS – *CAMPUS* BAMBUÍ-MG
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

JORGE LUÍS VIEIRA MURILO

**TUTORIAL PARA CONSTRUÇÃO DE OVA UTILIZANDO A FERRAMENTA
ROBLOX STUDIO**

**BAMBUÍ-MG
2023**

JORGE LUÍS VIEIRA MURILO

**TUTORIAL PARA CONSTRUÇÃO DE OVA UTILIZANDO A FERRAMENTA
ROBLOX STUDIO**

Tutorial elaborado para auxiliar professores que não possuem conhecimentos avançados em programação na construção de OVA personalizados utilizando o Roblox Studio.

**BAMBUÍ-MG
2023**

LISTA DE FIGURAS

Figura 1 - Botão de instalação da Ferramenta Roblox Studio.....	4
Figura 2 - Iniciar Instalação do Roblox Studio.....	5
Figura 3 - Etapa de Instalação.....	5
Figura 4 - Criar uma conta no Roblox.....	7
Figura 5 - Prenchimento de informações pessoais.....	8
Figura 6 - Confirmar o Login.....	9
Figura 7 - Tela do Roblox Studio.....	9
Figura 8 - Templates Básicos.....	10
Figura 9 - Criação de Experiência.....	11
Figura 10 - Login no Roblox Studio.....	12
Figura 11 - Salvar na Roblox.....	13
Figura 12 - Criar novo Jogo.....	13
Figura 13 - Informações Básicas.....	14
Figura 14 - Tela do Roblox Studio.....	14
Figura 15 - Tela do <i>website</i> Free3D.....	16
Figura 16 - Tela do <i>website</i> TurboSquid.....	17
Figura 17 - Importar modelo 3D no Roblox.....	18
Figura 18 - Importar modelo 3D no Roblox.....	19
Figura 19 - Importar Modelo.....	19
Figura 20 - Modelo Importado.....	20
Figura 21 - Exibição de janelas.....	22
Figura 22 - Seleção de um objeto “part”.....	23
Figura 23 - Adição de Script.....	23
Figura 24 - Testar a Experiência.....	24
Figura 25 - Exibição de janelas.....	25
Figura 26 - Repositório de Scripts, Haddock.....	26
Figura 27 - Repositório de Scripts, Haddock.....	26
Figura 28 - Local de Aparecimento do Jogador.....	28
Figura 29 - Inserção de objetos via Caixa de Ferramentas.....	28
Figura 30 - Inserção de objetos via Caixa de Ferramentas.....	29
Figura 31 - Cenário do OVA.....	30
Figura 32 - Inserção de cubo de gelo.....	31
Figura 33 - Inserção de retângulo.....	31
Figura 34 - Inserção de retângulo.....	32
Figura 35 - Mudança de Hierarquia de objetos.....	33
Figura 36 - Cubos.....	34
Figura 37 - Imagens na Caixa de Ferramentas.....	35
Figura 38 - Cubos com Texturas.....	36
Figura 39 - Inserir Objetos Script e ProximityPrompty.....	37
Figura 40 - Localizar ReplicatedStorage.....	39
Figura 41 - Quadro de Transformações.....	40
Figura 42 - Incluir ScreenGui.....	45
Figura 43 - Estrutura da Interface.....	46
Figura 44 - Estrutura da Interface.....	47
Figura 45 - Plano de Fundo ImageLabel.....	48
Figura 46 - Layout da experiência 2D.....	49

SUMÁRIO

1	COMO INSTALAR O ROBLOX STUDIO.....	4
1.1	Requisitos Mínimos para instalação do Roblox Studio.....	6
1.1.1	Requisitos do sistema operacional.....	6
1.1.2	Requisitos de hardware do sistema.....	6
2	RECURSOS DO ROBLOX STUDIO.....	7
2.1	Criar uma conta.....	7
2.2	Criar Projetos.....	10
2.2.1	Salvar na Roblox.....	10
2.3	Comandos Básicos.....	15
3	REPOSITÓRIOS DE MODELOS PRONTOS.....	16
3.1	Free3d.....	16
3.2	TurboSquid.....	17
3.3	Como Incluir Modelos no Projeto.....	17
4	SCRIPTS.....	21
4.1	Como e Onde Incluir Scripts.....	22
4.2	Repositórios de Scripts.....	25
5	COMO CRIAR UMA EXPERIÊNCIA 3D NO ROBLOX.....	27
6	CONSTRUÇÃO DE UMA EXPERIÊNCIA 2D.....	45

1 Como instalar o Roblox Studio

O Roblox Studio é uma ferramenta para construção de experiências personalizadas da plataforma Roblox, o qual está disponível no próprio site da empresa, acessível através do link: <https://create.roblox.com>. Para realizar a instalação dessa ferramenta, você deve acessar o link apresentado, e clicar no botão destacado pelo retângulo vermelho, **COMECE A CRIAR**, assim como demonstra a Figura 1.

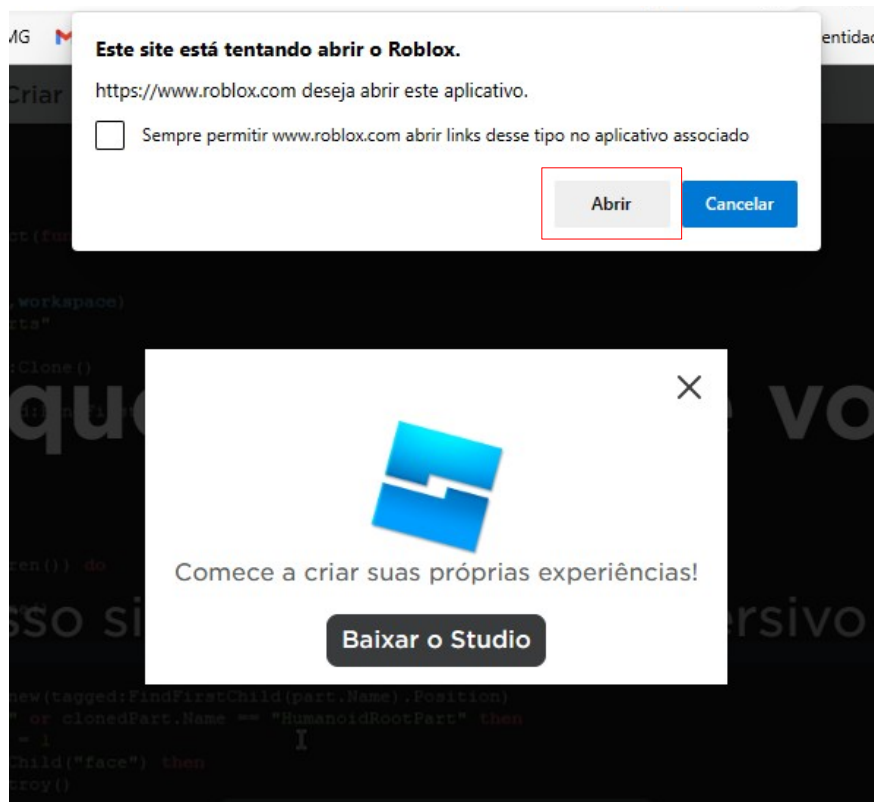
Figura 1 - Botão de instalação da Ferramenta Roblox Studio



Fonte: Os autores (2023).

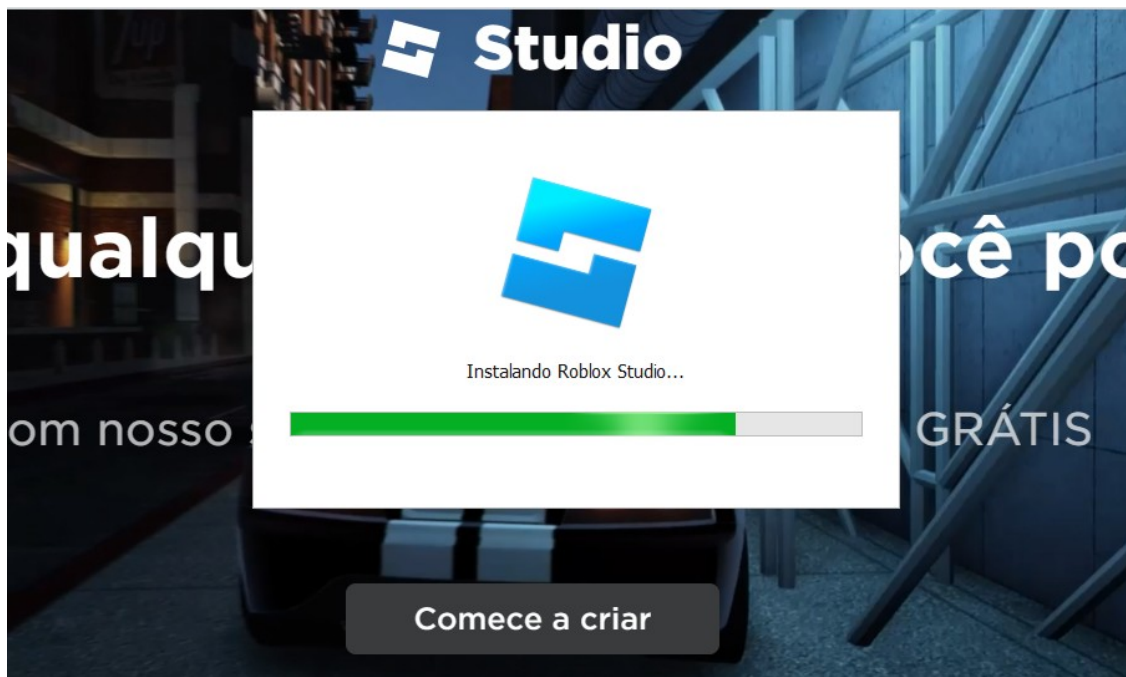
Após realizar o clique no botão **COMECE A CRIAR**, será exibida uma mensagem informando que o site está tentando abrir o Roblox, nesse momento, é necessário clicar no botão abrir, que será exibido em formato de janela de mensagem, conforme a Figura 2. Em seguida irá se iniciar a instalação do Roblox Studio, a qual se dará de forma automática, não sendo necessário realizar nenhuma configuração. As Figuras 2 e 3, demonstram esta etapa.

Figura 2 - Iniciar Instalação do Roblox Studio



Fonte: Os autores (2023).

Figura 3 - Etapa de Instalação



Fonte: Os autores (2023).

Após a realização dessas etapas a ferramenta Roblox Studio estará instalada em seu computador. Para que você utilize a ferramenta de forma adequada, recomenda-se que sua máquina possua os seguintes requisitos:

1.1 Requisitos Mínimos para instalação do Roblox Studio

Os requisitos mínimos do sistema para a Roblox estão listados nas seções a seguir.

1.1.1 Requisitos do sistema operacional

Para que seja possível utilizar as Ferramentas do Roblox, é necessário que o seu computador possua os seguintes requisitos do sistema operacional.

- **PC/Windows:** Windows 7, Windows 8.1 ou Windows 10. Para o Windows 8.1, é necessário executar o Roblox no modo Desktop, pois o modo Metro (a tela inicial lado a lado) não é suportado atualmente.
- **Navegares suportados pelo site do Roblox para PC:** Chrome, Firefox, e Microsoft Edge.
- **Mac:** O Roblox Client será instalado na versão 10.10 (Yosemite) e superior, enquanto o Roblox Studio será instalado no Mac OS 10.11 (El Capitan) e superior.
- **O site do Roblox suporta os navegadores para Mac:** Chrome, Firefox, e Safari.
- **Linux:** O Roblox não é suportada no Linux.

1.1.2 Requisitos de hardware do sistema

Para se obter uma boa performance, tanto no teste quanto na construção de experiências com o Roblox, é necessário que seu computador possua os seguintes requisitos mínimos de seus componentes:

- **Placa de vídeo:** No PC/Windows, o aplicativo Roblox requer suporte em nível de recurso DirectX 10 ou superior. Para obter o melhor desempenho, recomenda-se um computador com menos de 5 anos com placa de vídeo dedicada ou um laptop com menos de 3 anos com placa de vídeo integrada.
- **Processador:** Recomenda-se um processador recente (a partir do ano 2005) com uma velocidade de clock de 1,6 Ghz ou superior. Visto que houve alguns problemas com processadores AMD mais antigos
- **RAM ou Memória:** A Roblox recomenda que você tenha pelo menos 1 GB de memória no Windows 7, Windows 8.1 ou Windows 10.
- **Espaço de Armazenamento:** É recomendado que haja pelo menos 20 Mb de espaço de armazenamento do sistema para instalar o Roblox.

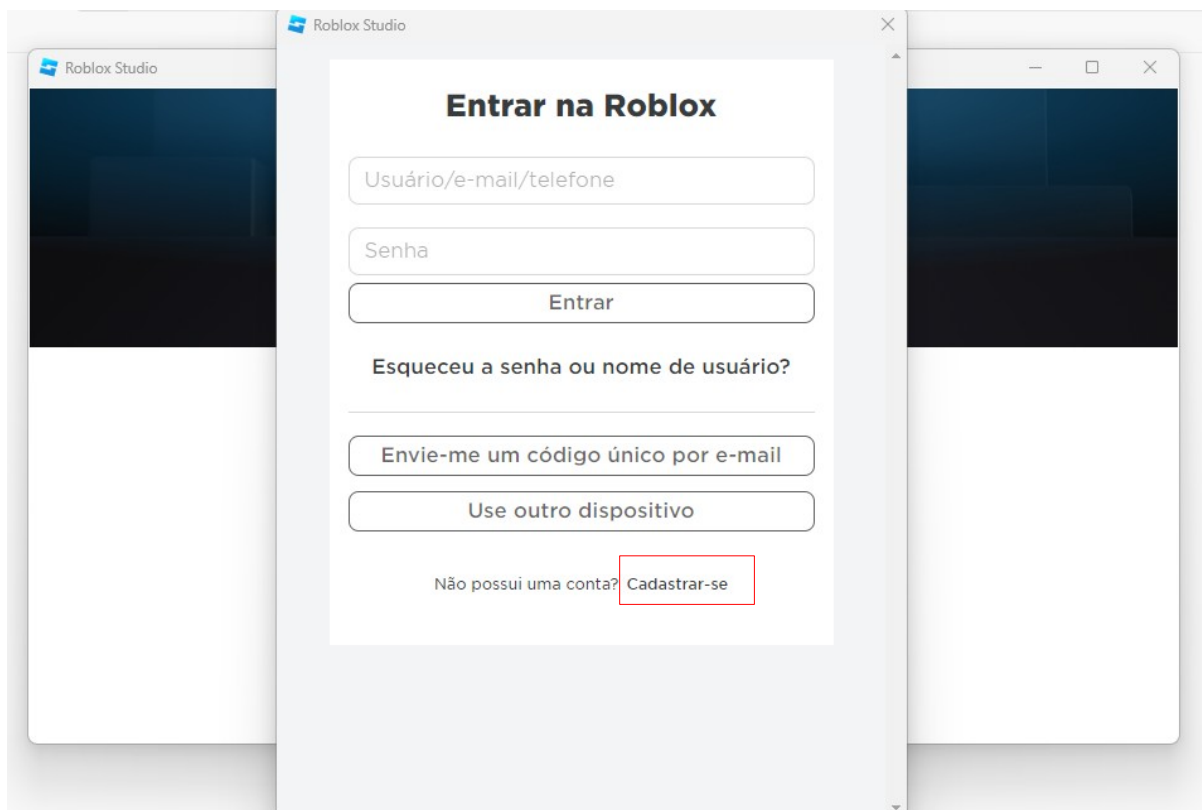
2 RECURSOS DO ROBLOX STUDIO

Para que seja possível acessar os recursos do Roblox Studio, é necessário que seja criada uma conta no Roblox, caso você ainda não a tenha criado. Do contrário, basta realizar *login* na sua conta e utilizar os recursos da Ferramenta.

2.1 Criar uma conta

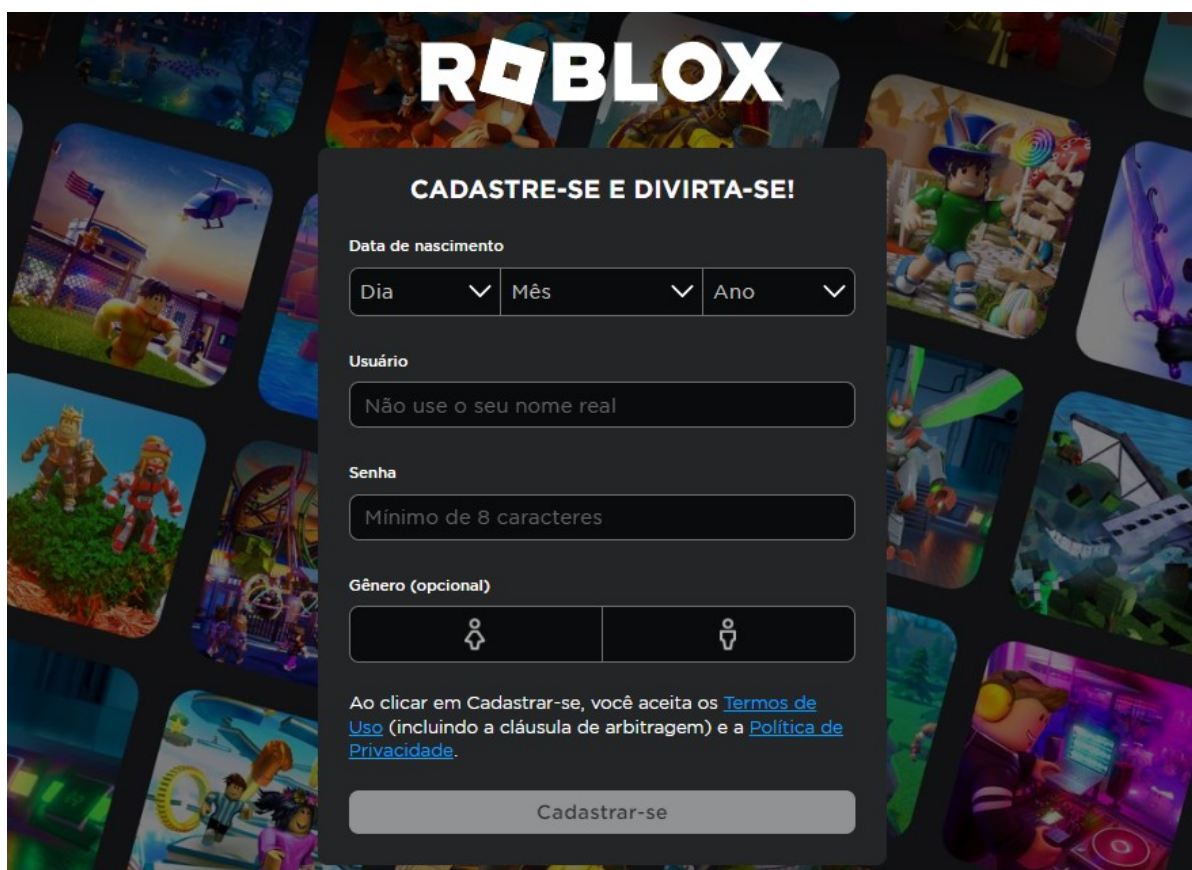
Caso ainda não tenha uma conta, basta criá-la clicando no link **Cadastre-se** do Roblox Studio, o qual estará disponível na tela durante o seu primeiro acesso à ferramenta, conforme exemplificado na Figura 4. Após clicar no link, você será redirecionado para uma tela semelhante à demonstrada na Figura 5.

Figura 4 - Criar uma conta no Roblox



Fonte: O autor (2023).

Figura 5 - Preenchimento de informações pessoais



The image shows the Roblox registration form overlaid on a background collage of various Roblox game thumbnails. The form is titled "CADASTRE-SE E DIVIRTA-SE!" and contains the following fields and options:

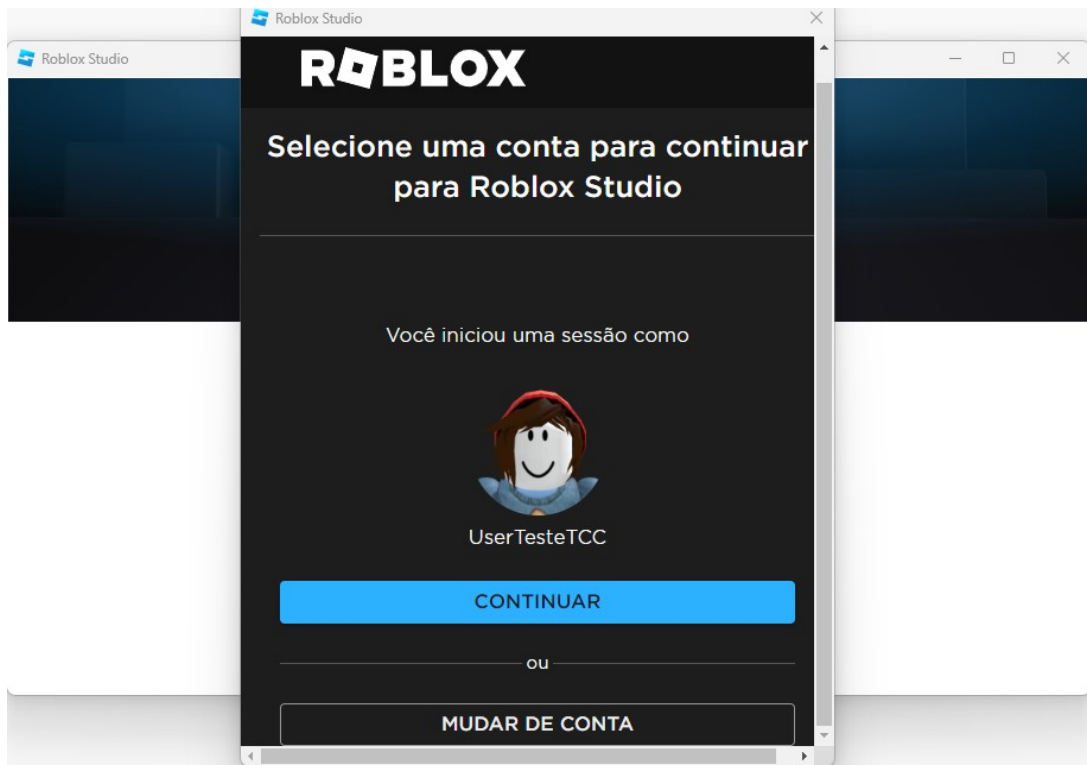
- Data de nascimento**: Three dropdown menus for "Dia", "Mês", and "Ano".
- Usuário**: A text input field with the placeholder text "Não use o seu nome real".
- Senha**: A text input field with the placeholder text "Mínimo de 8 caracteres".
- Gênero (opcional)**: Two radio button options represented by male and female icons.
- Legal Notice**: A line of text stating "Ao clicar em Cadastrar-se, você aceita os [Termos de Uso](#) (incluindo a cláusula de arbitragem) e a [Política de Privacidade](#)."
- Button**: A large button at the bottom labeled "Cadastrar-se".

Fonte: O autor (2023).

Após a finalização do cadastro você será redirecionado para a página inicial da plataforma Roblox. Nesse momento você poderá fechar o seu navegador e realizar login no Roblox Studio, assim como evidenciado na Figura 5. Para realizar o login você deverá preencher o campo “usuário” como nome cadastrado na plataforma e preencher sua senha, para que seja possível acessar as funcionalidades da ferramenta. Depois disso você deve clicar no botão Entrar.

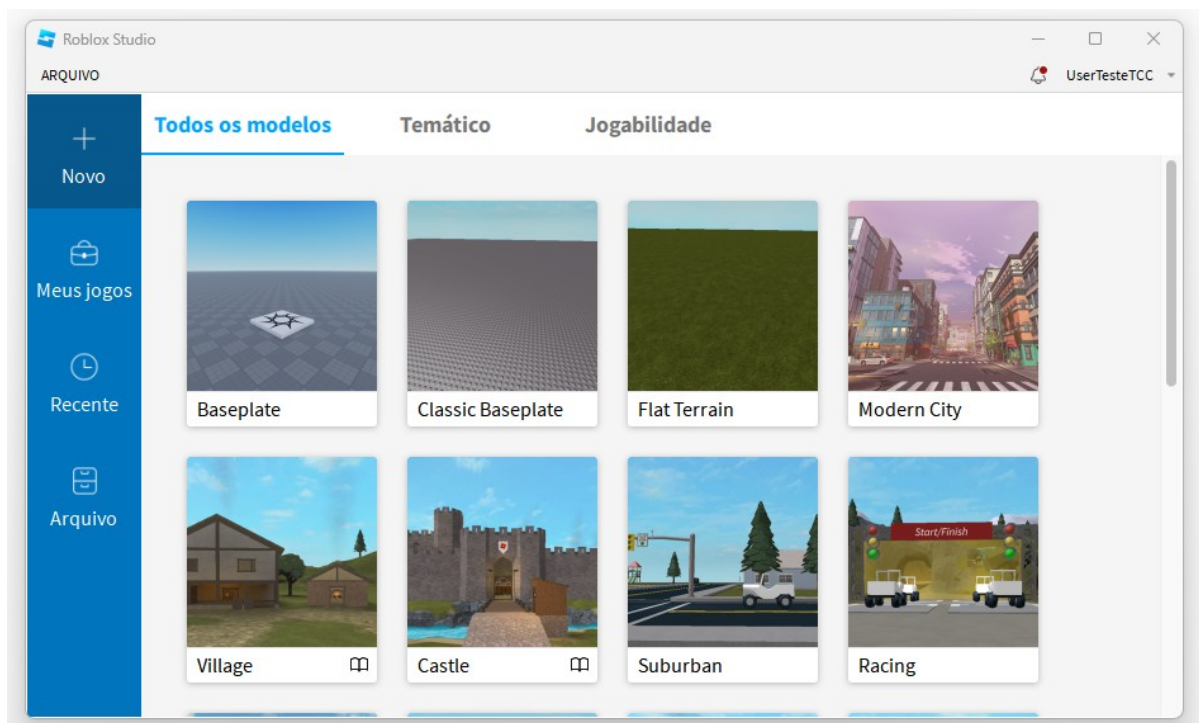
Na tela evidenciada na Figura 6, é apresentado o passo seguinte, em que você deverá clicar no botão continuar. Pronto! Você agora tem acesso aos recursos do Roblox Studio e verá uma tela semelhante à demonstrada na Figura 7.

Figura 6 - Confirmar o Login



Fonte: O autor (2023).

Figura 7 - Tela do Roblox Studio

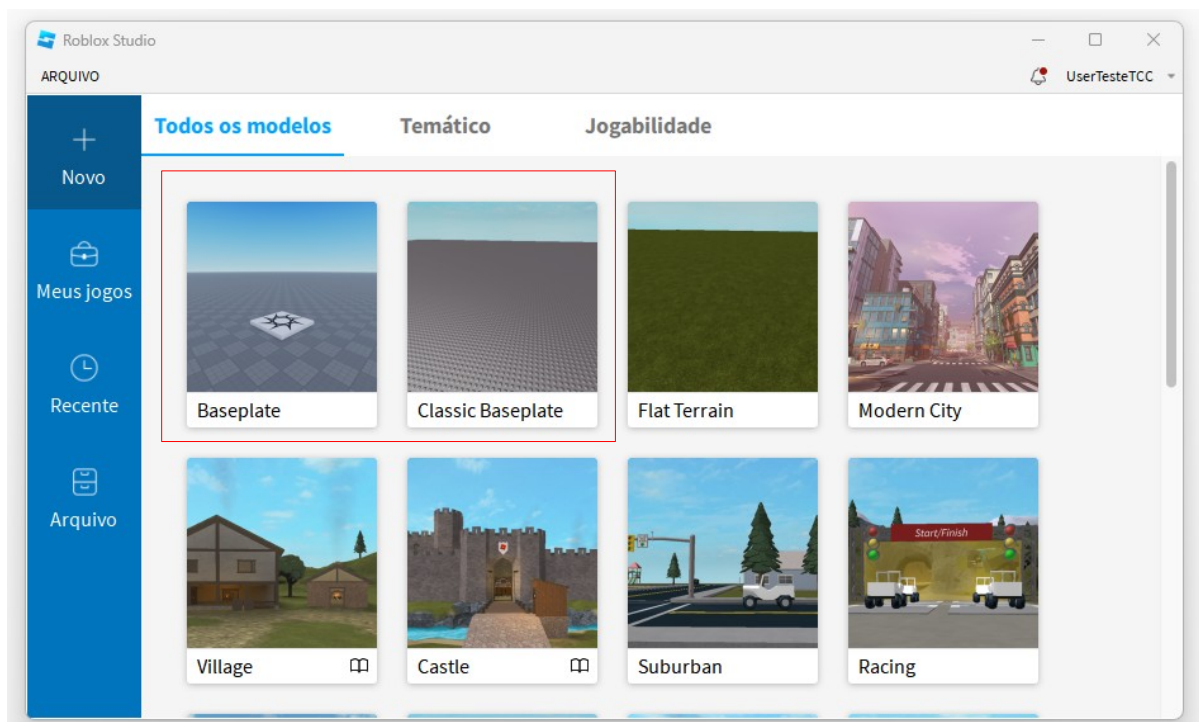


Fonte: O autor (2023).

2.2 Criar Projetos

Para criar um projeto, existem diversos modelos prontos, os quais incluem cenários diversificados. Dentre eles, existem dois que constituem os modelos mais básicos para construção de experiências, e podem ser utilizados para construir experiências **do zero**. Esses, encontram-se destacados pelo retângulo vermelho na Figura 8.

Figura 8 - Templates Básicos



Fonte: O autor (2023).

A diferença entre esses cenários é quanto a possibilidade de escolha do local onde o personagem surgirá. No primeiro modelo, chamado de **Baseplate**, é possível realocar a estrutura de surgimento de acordo com a vontade do desenvolvedor da experiência, já no segundo modelo, este surgirá em um local padrão do cenário.

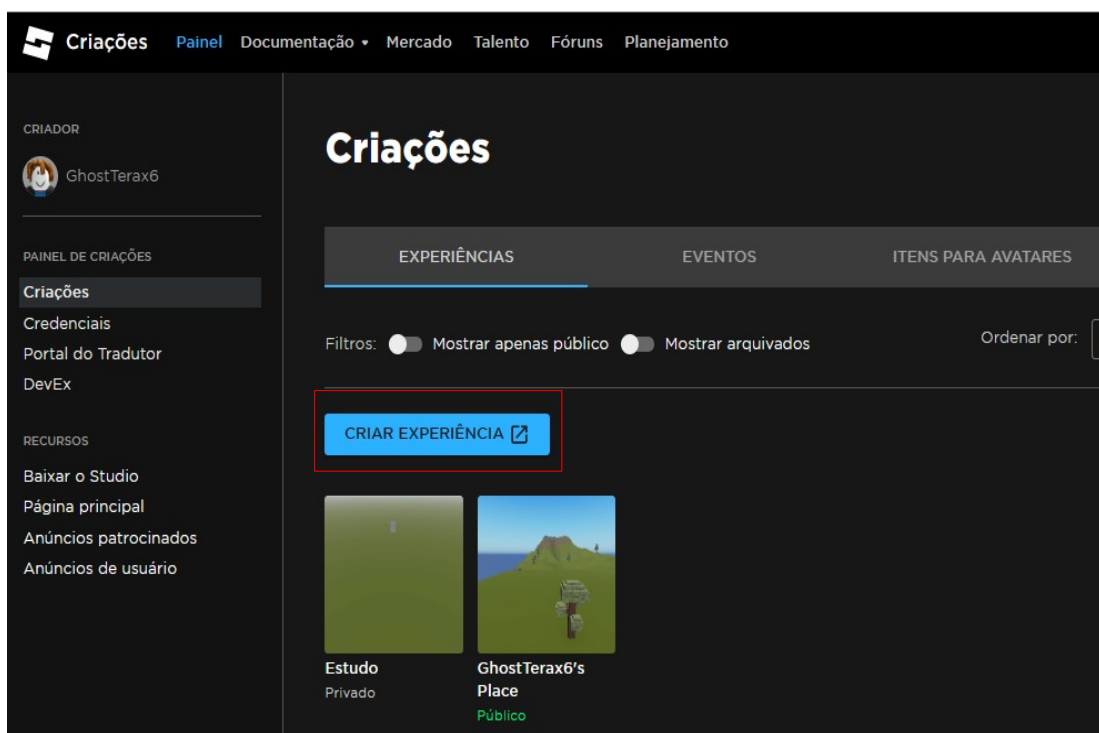
2.2.1 Salvar na Roblox

Para assegurar que seus modelos não sejam perdidos e para facilitar o acesso a todos os recursos da plataforma, é altamente recomendável salvar suas criações diretamente no site. Ao fazer isso, você poderá usufruir de diversas funcionalidades de maneira mais fácil, incluindo a adição de paisagens, imagens clicáveis e objetos de interface.

Antes de mais nada, é necessário criar suas experiências, para isso existem duas formas de se criá-las. Uma delas foi demonstrada nas sessões anteriores, e a segunda pode ser realizada ao se acessar o link <https://create.roblox.com/dashboard/creations>. Caso você não tenha feito no login na plataforma, você deve clicar no botão **Entrar na Conta**, que se encontra na mesma posição do botão evidenciado no canto superior direito da Figura 1.

Após clicar no botão, você deve preencher seu usuário e senha, assim como demonstrado na Figura 4. Após realizar o *login* na plataforma, uma tela semelhante a demonstrada na Figura 9, será exibida. Nessa tela, você deve clicar no botão **CRIAR EXPERIÊNCIA**, destacado na imagem pelo retângulo vermelho.

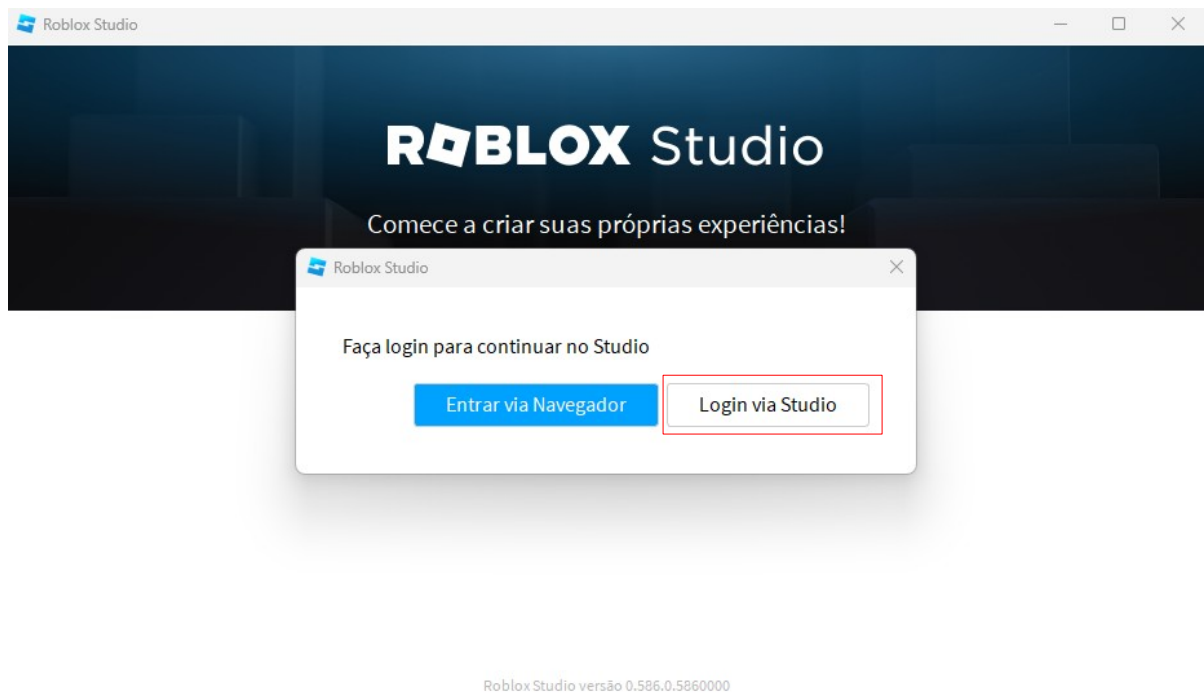
Figura 9 - Criação de Experiência



Fonte: O autor (2023).

Clicando-se no botão destacado, uma mensagem semelhante à demonstrada na Figura 2 será exibida. Dessa forma você deve clicar em abrir, caso você não tenha instalado o Roblox Studio ainda, retorne à sessão 1 e realize a instalação. Após clicar no botão, o Roblox Studio abrirá, e caso você não tenha realizado login durante a instalação dele, uma tela semelhante à demonstrada na Figura 10 será exibida.

Figura 10 - Login no Roblox Studio

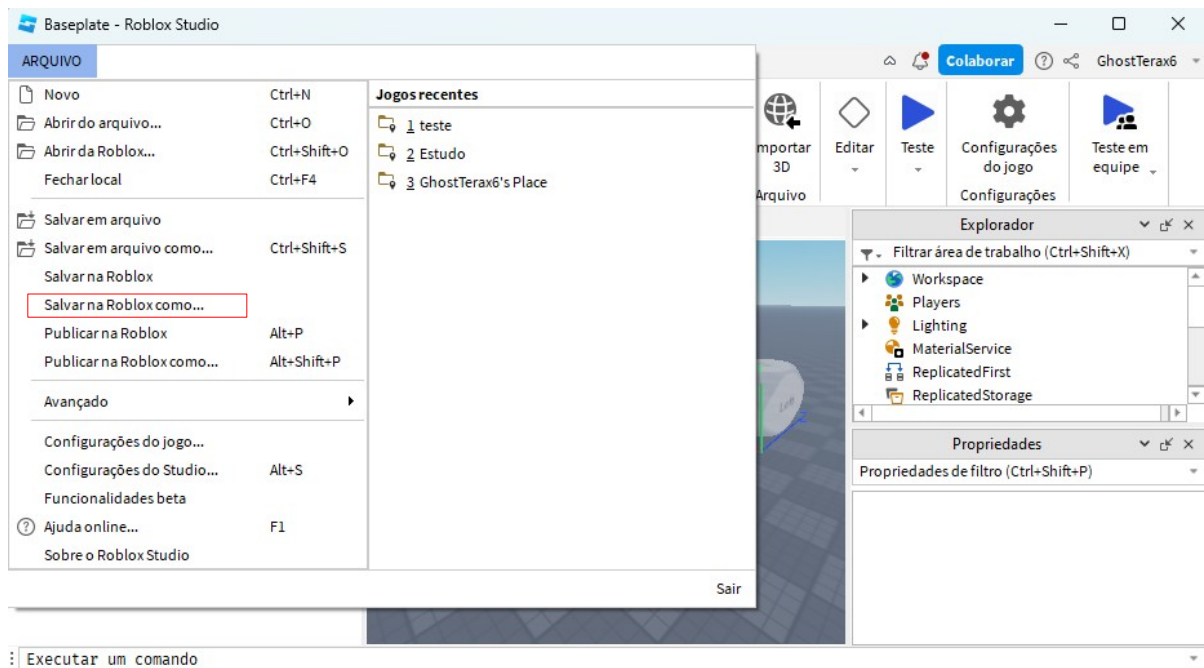


Fonte: O autor (2023).

Para garantir um login com sucesso, selecione **Login via Studio**, destacado pelo retângulo vermelho. Você deverá seguir os passos evidenciados na Figura 4 e em seguida selecionar o *template* de sua escolha.

Ambas as formas de se criar experiências possuem a mesma eficiência, basta escolher a que melhor se adapta. Porém o ponto mais importante dessa sessão, é destacar a necessidade de se salvar a experiência criada na Roblox, para isso você deve selecionar no canto superior esquerdo o menu **Arquivo**, o qual exibirá uma lista de opções, assim como demonstra a Figura 11. Nesta lista você deve clicar na opção **Salvar na Roblox como...**, destacada na figura com o retângulo vermelho.

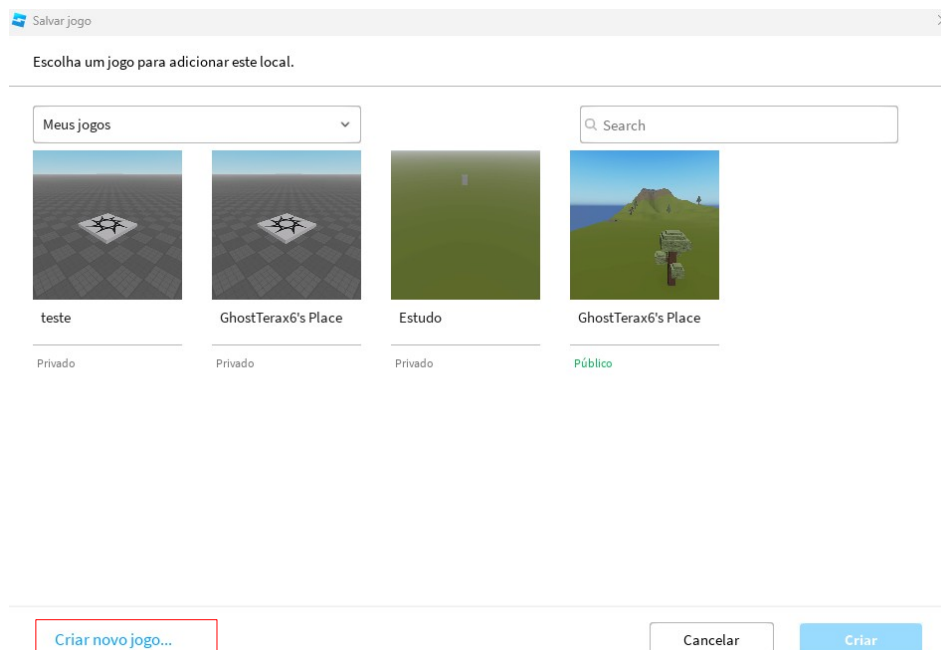
Figura 11 - Salvar na Roblox



Fonte: O autor (2023).

Em seguida a tela evidenciada pela Figura 12, será exibida. Nesta tela, você deve clicar em **Criar novo jogo....**

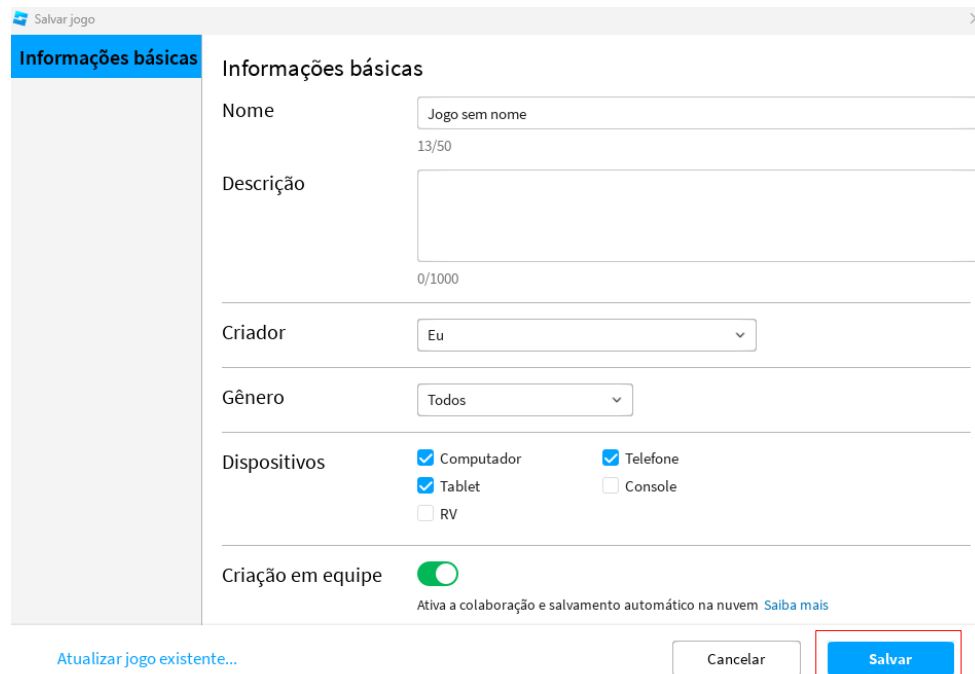
Figura 12 -Criar novo Jogo



Fonte: O autor (2023).

Em seguida, preencha as informações básicas da sua experiência e clique em salvar.

Figura 13 -Informações Básicas



Salvar jogo

Informações básicas

Nome: Jogo sem nome (13/50)

Descrição: (0/1000)

Criador: Eu

Gênero: Todos

Dispositivos:

- ☒ Computador
- ☒ Telefone
- ☒ Tablet
- ☐ Console
- ☐ RV

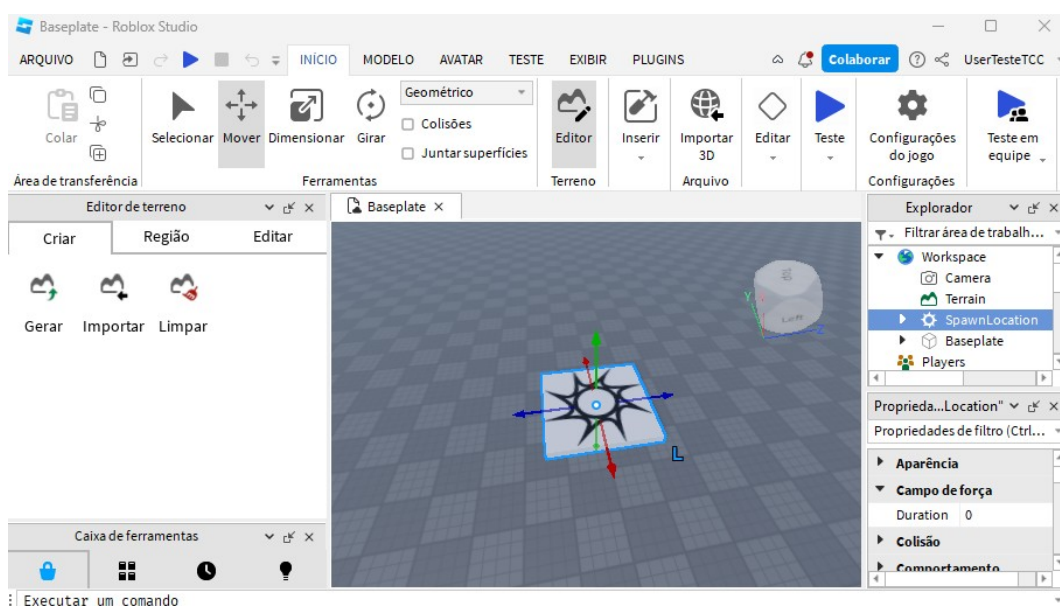
Criação em equipe: ☒ Ativa a colaboração e salvamento automático na nuvem [Saiba mais](#)

[Atualizar jogo existente...](#) Cancelar Salvar

Fonte: O autor (2023).

Após isso, você visualizará uma tela semelhante à da Figura 14, em que você poderá construir sua experiência personalizada.

Figura 14 - Tela do Roblox Studio



Fonte: O autor (2023).

2.3 Comandos Básicos

O Roblox Studio, possui alguns atalhos para auxiliar na construção das experiências, e facilitar a manipulação do ambiente, dentre eles pode-se destacar as teclas:

- W, A, S, D: Movimenta a câmera para frente, esquerda, para trás e para direita, respectivamente;
- Q, E: Movimenta a câmera para baixo e para cima respectivamente;
- Shift: Diminui a velocidade de movimentação da câmera ao ser pressionado em conjunto com algum botão de movimento, por exemplo, shift + W.
- F: Movimenta a câmera instantaneamente para o objeto selecionado, ou seja, centraliza o objeto no ambiente de visualização.

Além das teclas de atalho, o mouse também auxilia na manipulação dos objetos para construção da experiência. Seus botões possuem as seguintes funções:

- Botão Direito: Quando pressionado permite rotacionar a visualização do ambiente ao redor, ou seja, muda a perspectiva de visualização do ambiente de desenvolvimento.
- Scroll: Também conhecida como “rodinha” do mouse, ao ser rotacionada, permite aumentar ou diminuir o zoom, permitindo visualizar detalhes com maior nitidez.
- Pressionar o Scroll: Pressionar o Scroll e movimentar o mouse, permite arrastar a câmera, sem que haja necessidade de pressionar os botões de movimentação.

3 Repositórios de Modelos Prontos

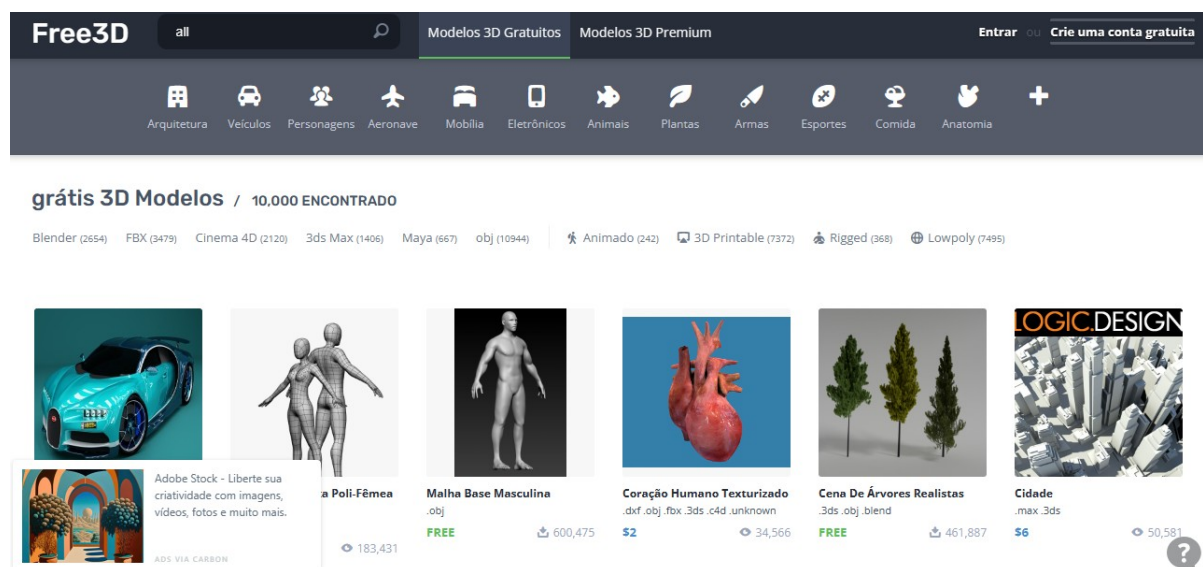
O Roblox studio suporta objetos 3D no formato **.fbx** e **.obj**, esses sufixos de arquivos, constituem uma espécie de rótulo que descreve o tipo ou o formato de um arquivo. Ele é uma convenção utilizada em sistemas operacionais para facilitar a abertura desse arquivo pelo programa que o gerou.

Suponhamos que um arquivo de texto tenha o nome **Teste**, ao observar a maneira como esse arquivo está disponível em seu computador você verá algo como **Teste.txt**, o que facilitará para o sistema na identificação de qual programa é o adequado para abrí-lo. Isso não é diferente para modelos 3D, que após seu download, terão nomes como **modelo3D.fbx**. Esses modelos podem ser encontrados em diversas plataformas, e algumas delas encontram-se nas sessões a seguir.

3.1 Free3d

O Free3D, trata-se de um repositório com uma grande variedade de modelos 3D prontos para download. O site possui mais de 10.000 modelos gratuitos, que vão desde arquitetura à anatomia. Nessa plataforma é possível encontrar modelos em diversos formatos, dentre eles os suportados pelo Roblox Studio. Assim como evidenciado na Figura 15.

Figura 15 - Tela do *website* Free3D



Fonte: O autor (2023).

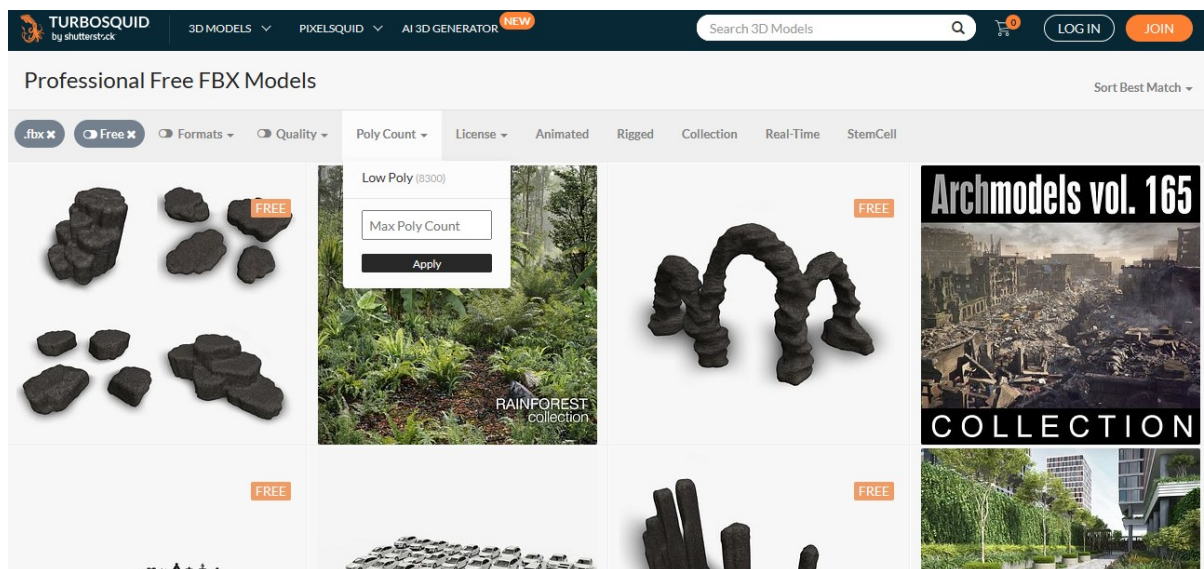
O Free3D, pode ser acessado através do link: <https://free3d.com/pt/>. Caso você tenha

afinidade com a construção de modelos 3D, esse *website* possibilita a disponibilização de modelos construídos pela comunidade externa, viabilizando a disponibilização de seu modelo para outros desenvolvedores de experiências com modelos 3D.

3.2 TurboSquid

Semelhante ao Free3D, o TurboSquid também é um repositório de modelos 3D. Esse *website* possui tanto modelos pagos quanto modelos gratuitos, mas diferencia-se do Free3D pelo filtro de quantidade de polígonos dos modelos. A quantidade de polígonos influencia em quão detalhado é o objeto 3D, porém quanto mais polígonos, maior o custo computacional necessário para manipular esse objeto, tanto na etapa de construção quanto na de utilização da experiência. O Roblox Studio suporta modelos com até 15.000 polígonos. A Figura 16 ilustra a aparência desse repositório de modelos 3D.

Figura 16 - Tela do *website* TurboSquid



Fonte: O autor (2023).

O repositório pode ser acessado através do link: <https://www.turbosquid.com/>

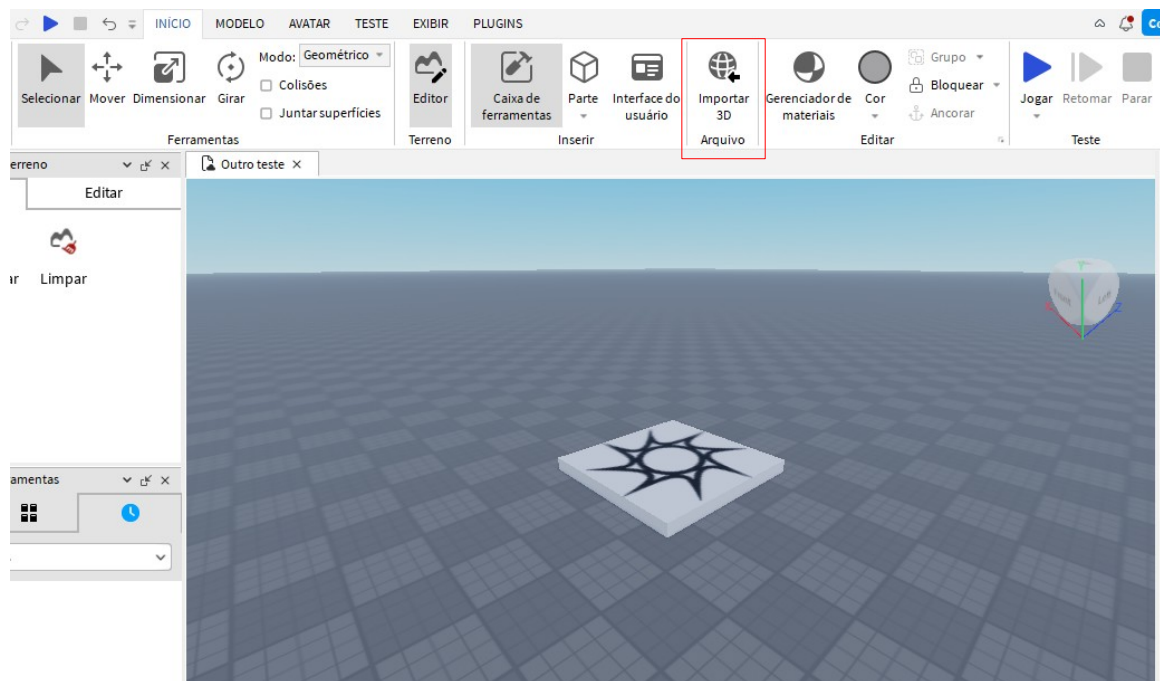
3.3 Como Incluir Modelos no Projeto

Para incluir modelos prontos no projeto, você deve acessar algum repositório, ou construir seu próprio modelo, respeitando o padrão aceito pelo Roblox Studio, o .fbx ou .obj. Utilizando-se modelos prontos, você pode acessar algum dos repositórios descritos na sessão anterior e buscar pelo item que deseja.

Para incluir o modelo na sua experiência do Roblox, basta clicar no botão **Importar 3D** do menu de ferramentas do Roblox Studio. Assim como demonstrado na Figura 17. Em seguida, basta localizar o modelo em seu computador e clicar no botão abrir, conforme a Figura 18.

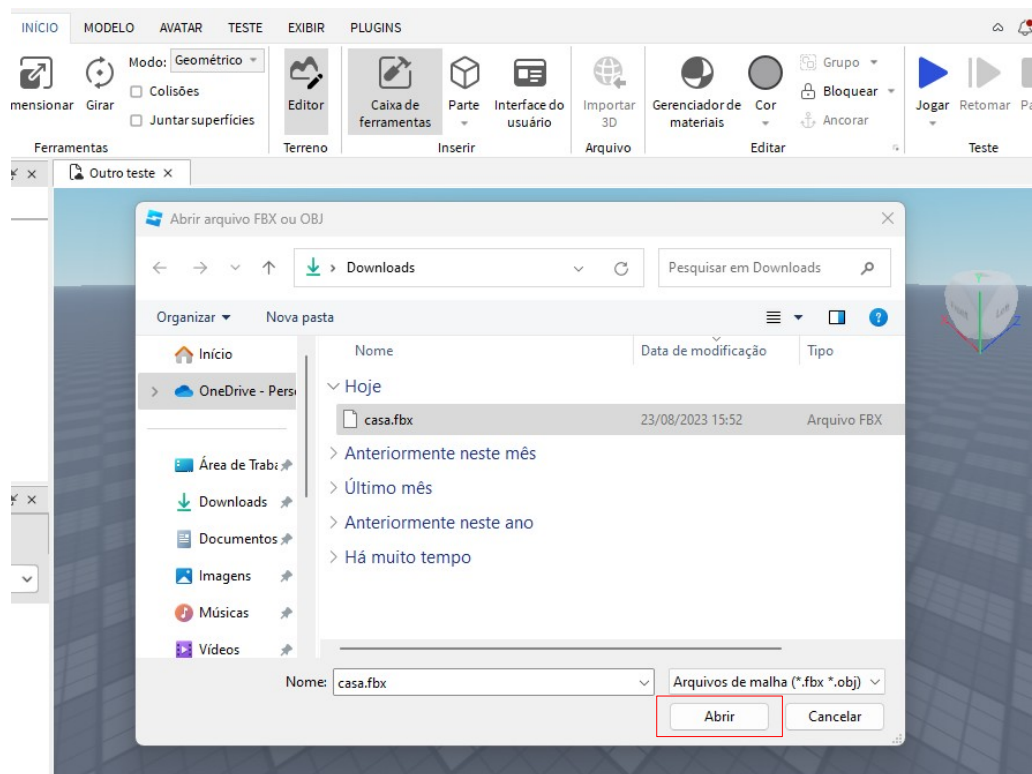
Após Clicar no botão abrir será exibida uma Tela Conforme a Figura 19, em que se deve clicar no botão Importar, destacado pelo retângulo vermelho.

Figura 17 - Importar modelo 3D no Roblox



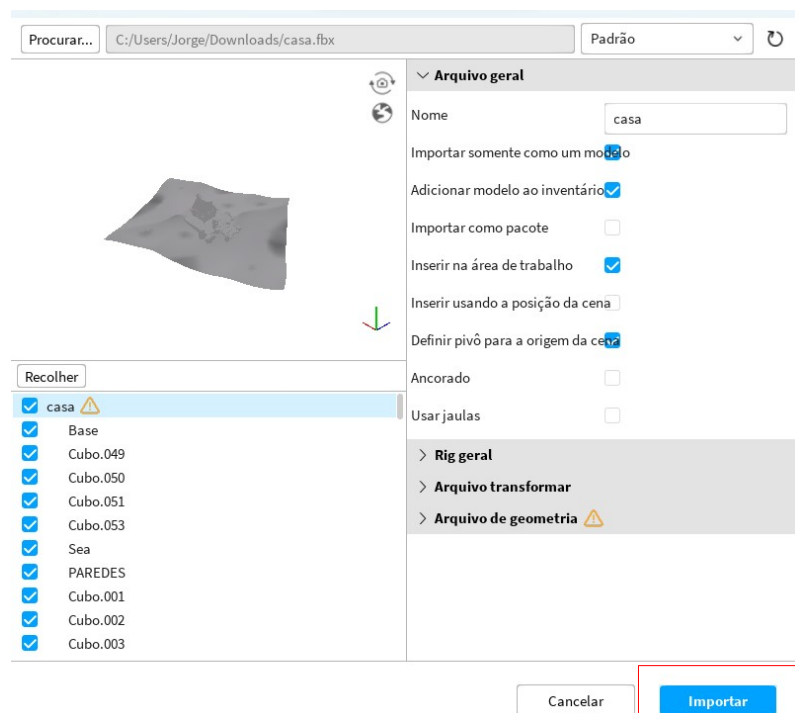
Fonte: O autor (2023).

Figura 18 - Importar modelo 3D no Roblox



Fonte: O autor (2023).

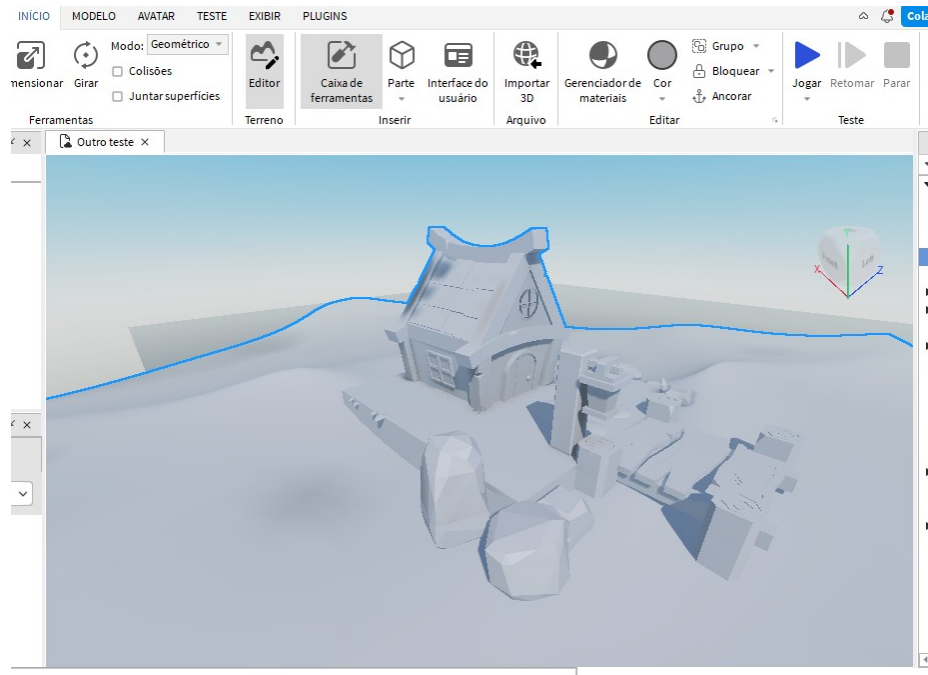
Figura 19 - Importar Modelo



Fonte: O autor (2023).

Após a importação, será possível visualizar o modelo e manipulá-lo. Assim como demonstra a Figura 20.

Figura 20 - Modelo Importado



Fonte: O autor (2023).

4 Scripts

No contexto do Roblox Studio, scripts são trechos de código escritos em Lua, uma linguagem de programação, que são usados para controlar o comportamento de objetos, personagens e cenários dentro de experiências criados na plataforma Roblox. Os *scripts* são uma parte essencial do desenvolvimento no Roblox, pois permitem aos desenvolvedores criar interatividade e funcionalidades específicas para seus jogos.

Dentre as principais características dos *scripts* no Roblox Studio têm-se:

- **Lua:** Roblox Studio utiliza a linguagem de programação Lua para escrever *scripts*. Lua é uma linguagem versátil e é frequentemente usada em jogos devido à sua simplicidade.
- **Eventos e Funções:** Os *scripts* no Roblox Studio frequentemente respondem a eventos, como o clique do mouse ou a colisão de objetos. Eles também podem conter funções que são chamadas em momentos específicos para realizar ações desejadas.
- **Manipulação de Objetos:** Os *scripts* podem ser usados para controlar o movimento, a aparência e o comportamento de objetos no jogo, como personagens, veículos, itens e entre outros.
- **Lógica de Jogo:** Os *scripts* são essenciais para implementar a lógica do jogo, incluindo mecânicas de jogabilidade, regras, pontuação e outras funcionalidades específicas do jogo.
- **Servidores e Clientes:** No contexto de jogos multiplayer no Roblox, é comum ter *scripts* no lado do servidor (Server Scripts) e scripts no lado do cliente (Local Scripts). Os *scripts* no lado do servidor lidam com lógica de jogo que precisa ser executada no servidor, enquanto os *scripts* no lado do cliente lidam com a interação do jogador e a exibição de informações.

Os desenvolvedores do Roblox usam o Roblox Studio para criar e editar *scripts*, incorporando-os em seus jogos para criar experiências interativas e envolventes. O Roblox Studio fornece uma interface gráfica intuitiva para criar e manipular *scripts*, tornando o processo de desenvolvimento acessível a uma ampla gama de usuários, desde iniciantes até desenvolvedores mais experientes.

4.1 Como e Onde Incluir Scripts

No Roblox Studio, você pode incluir *scripts* para controlar o comportamento de objetos, personagens e outros elementos do seu jogo. Para isso você deve incluí-los no Roblox Studio da seguinte maneira:

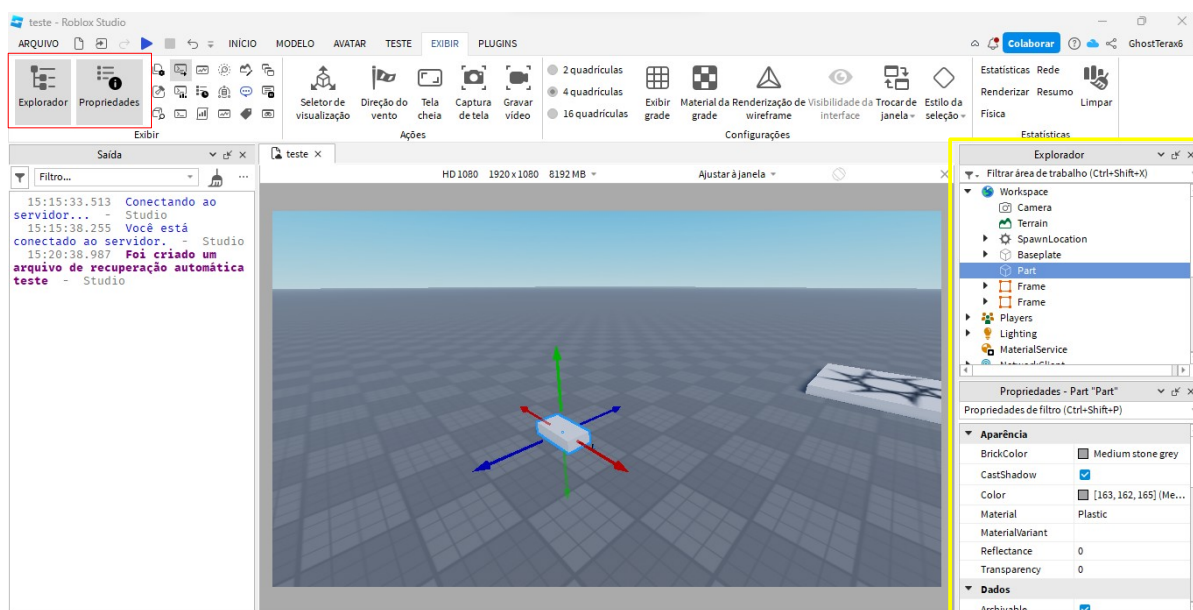
1. Abrir o Roblox Studio:

Inicie o Roblox Studio no qual você deseja incluir *scripts*.

2. Abrir a Janela Explorador e Propriedades:

Certifique-se de ter as janelas **Explorador** e **Propriedades** abertas. Você pode encontrá-las na barra de menus **Exibir**, assim como demonstra o retângulo vermelho na Figura 21. Essas funcionalidades estarão abertas se o plano de fundo das janelas estiver com a cor acinzentada, e será possível visualizá-las na lateral direita do ambiente, assim como destaca o retângulo amarelo.

Figura 21 - Exibição de janelas

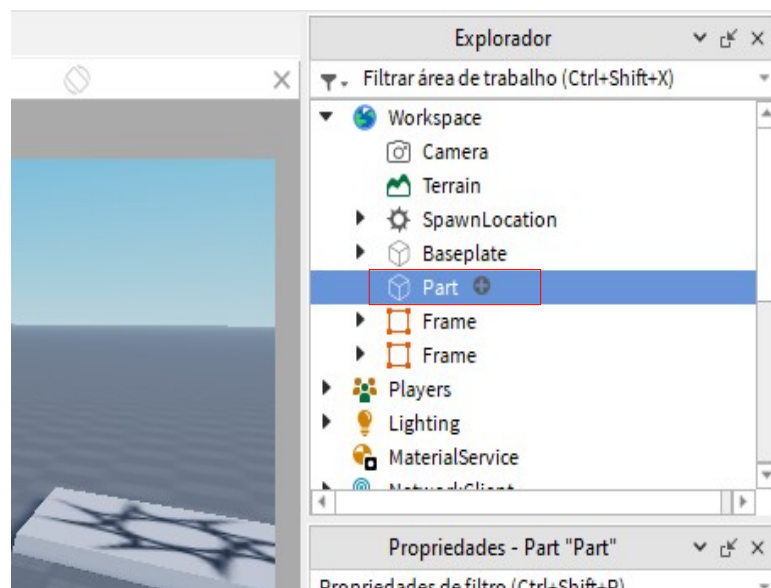


Fonte: O autor (2023).

3. Selecionar o Objeto alvo:

No **Explorador**, selecione o objeto ao qual deseja adicionar o *script*. Ele pode ser um bloco (part), um personagem, um veículo, ou qualquer outro objeto no seu jogo. Ao selecioná-lo, sua cor de fundo ficará azulada.

Figura 22 - Seleção de um objeto “part”

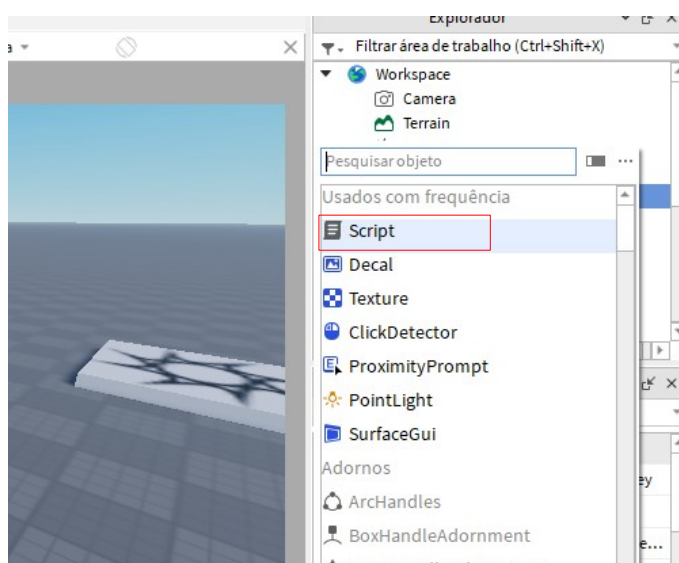


Fonte: O autor (2023).

4. Adicionar um Script:

Com o objeto selecionado, e o cursor sobre o seu nome, um ícone de + irá aparecer ao lado direito do objeto, clique sobre o + e escolha **Script** na lista. Assim como demonstra a Figura 23.

Figura 23 - Adição de Script

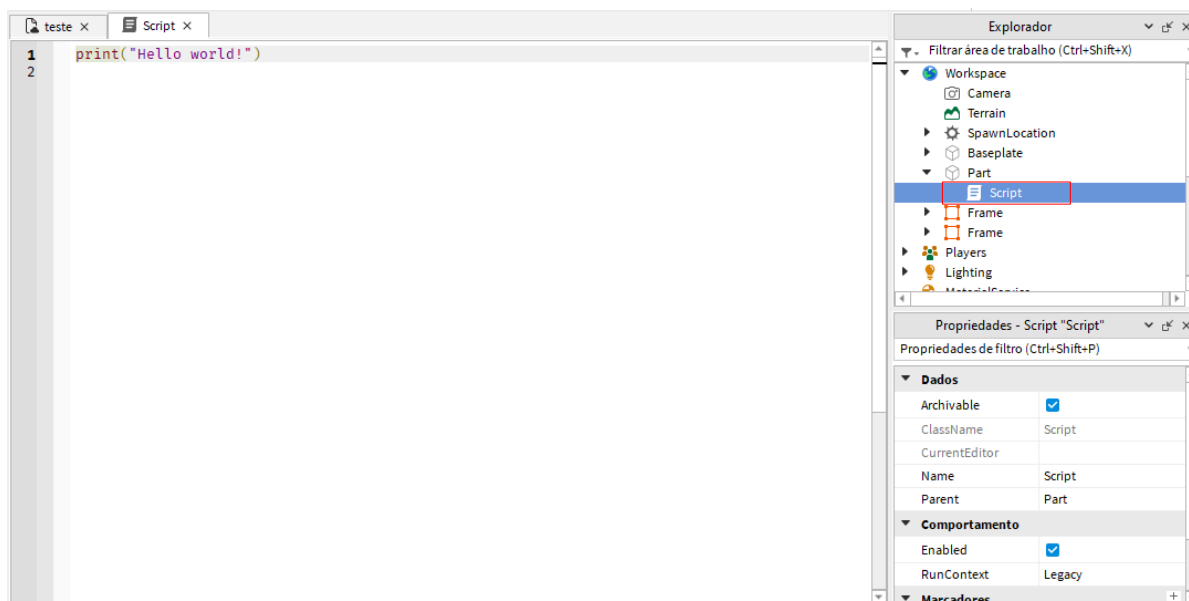


Fonte: O autor (2023).

5. Editar o Script:

Ao adicionar o *script*, uma nova guia chamada **Script** aparecerá no **Explorador**. Clique duas vezes no *script* para abrir o editor de *scripts*, onde você pode escrever ou colar seu código Lua.

Figura 24 - Testar a Experiência



Fonte: O autor (2023).

6. Escrever o Código Lua:

Dentro do editor de *scripts*, você pode começar a escrever seu código Lua. Este código será executado quando o jogo estiver em execução e o evento associado ao *script* for acionado.

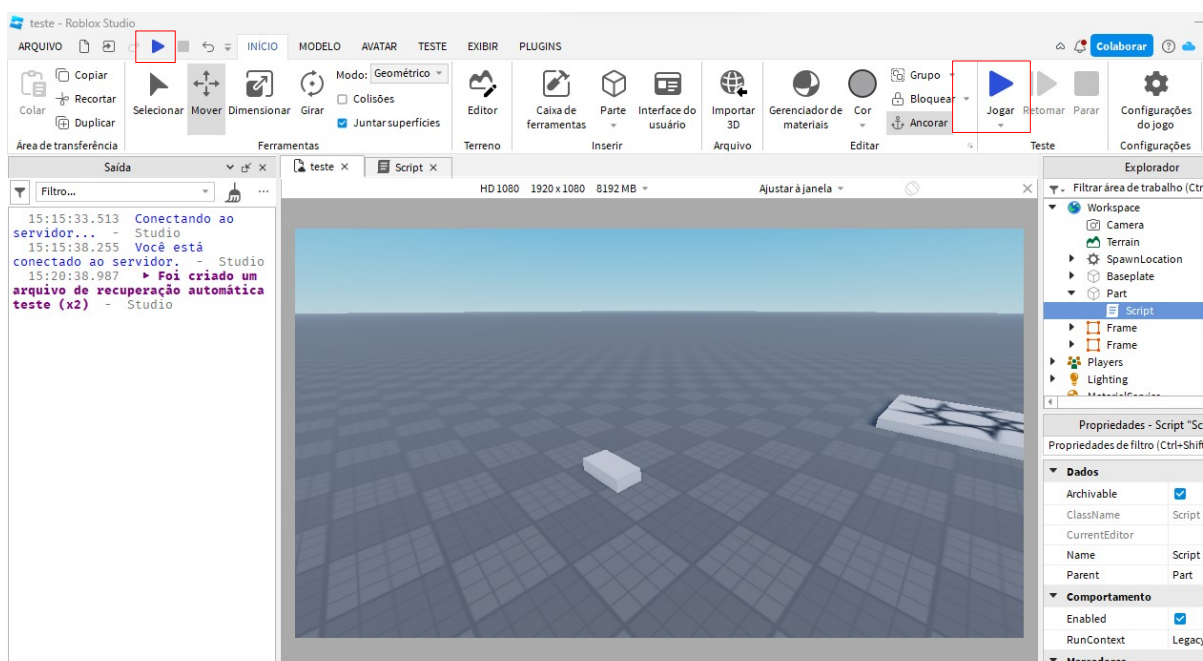
7. Salvar e Testar:

Certifique-se de salvar o lugar após adicionar ou editar *scripts*. Você pode fazer isso clicando em **Arquivo** e selecionando **Salvar**.

Para testar o *script*, clique em **Jogar** na barra de menus para iniciar a execução do jogo e observar o comportamento do *script*, assim como demonstra os retângulos vermelhos na Figura 25. Lembre-se de que, dependendo do que você deseja alcançar, pode ser necessário adicionar *scripts* em diferentes locais. Além disso, os *scripts* podem ser associados a eventos específicos para controlar a lógica do jogo de maneira mais dinâmica.

Existem dois botões para executar a experiência no menu **Início**, os quais possuem a mesma funcionalidade e resultado.

Figura 25 - Exibição de janelas



Fonte: O autor (2023).

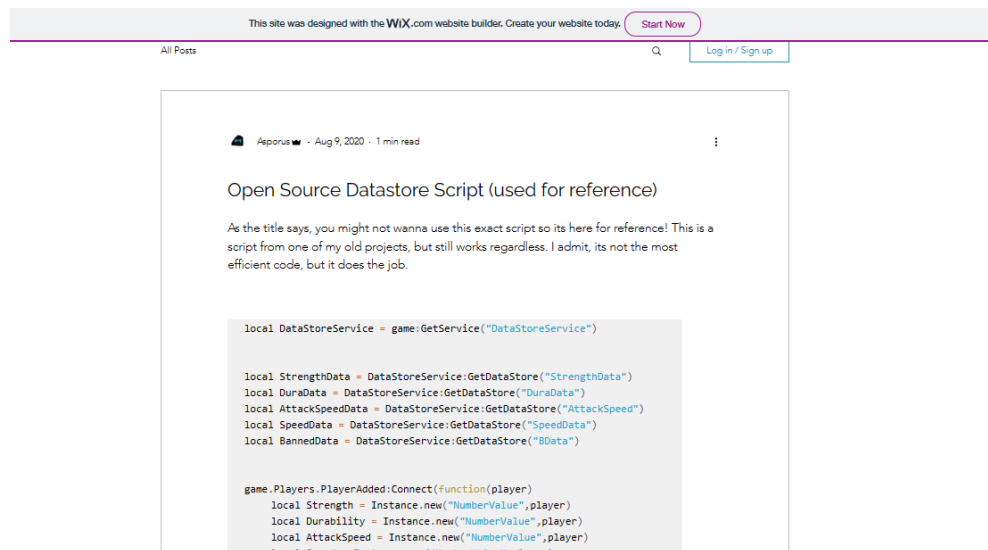
4.2 Repositórios de Scripts

Para facilitar o processo de escrita de *scripts*, existem atualmente dois repositórios que oferecem exemplos de código. Um deles é a Open Source Library | Asporus, que está em inglês e inclui seis exemplos de *scripts*.

O segundo repositório é o Haddock - Roblox Library, que disponibiliza um total de cento e dois exemplos de scripts de forma gratuita. O diferencial do Haddock em comparação com o Asporus é a presença de um chat interativo, baseado no ChatGPT 4.0, com foco exclusivo na geração de *scripts* para o Roblox Studio. Porém, esse chat é pago. As Figuras 26 e 27 ilustram esses repositórios.

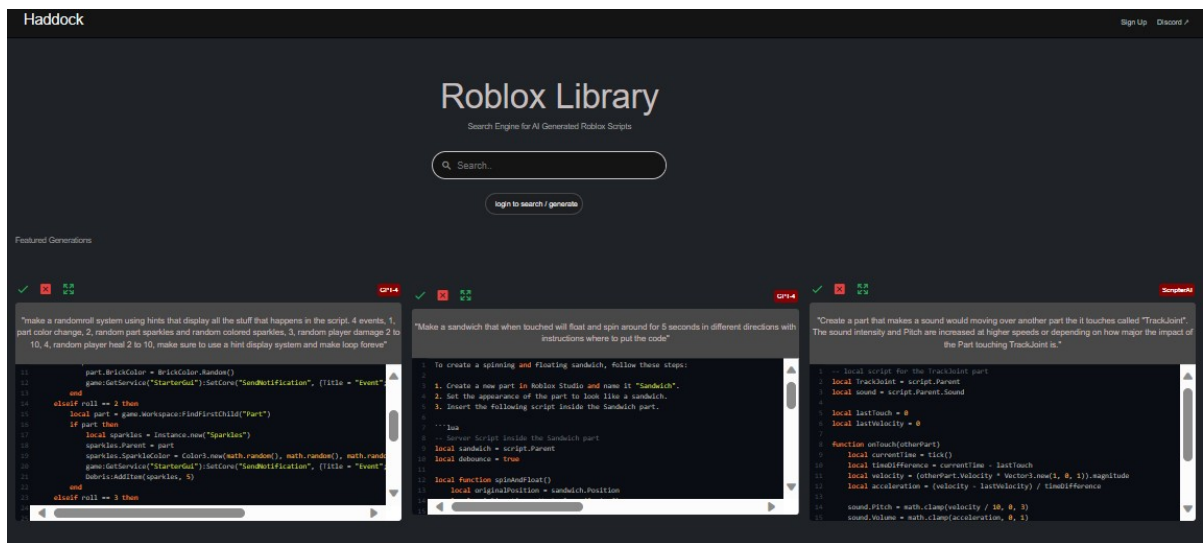
Para acessar o Asporus, visite [Open Source Library | Asporus \(licksaw23.wixsite.com\)](https://licksaw23.wixsite.com/asporus). Para explorar o Haddock - Roblox Library, acesse [Haddock - Roblox Library](https://haddock-roblox-library.com). Esses recursos são valiosos para quem busca aprimorar suas habilidades de scripting no Roblox Studio.

Figura 26 - Repositório de Scripts, Haddock



Fonte: O autor (2023).

Figura 27 - Repositório de Scripts, Haddock



Fonte: O autor (2023).

5 Como Criar uma Experiência 3D no Roblox

A experiência descrita a seguir trata-se de uma exploração do Roblox Studio como ferramenta para a construção de OVA (Objetos Virtuais de Aprendizagem) e aborda o uso dessa plataforma como meio de ensinar o conceito de mudanças de estado físico.

Ao empregar o Roblox Studio para criar OVAs, é possível proporcionar uma abordagem prática e envolvente para a compreensão de conceitos complexos, como as mudanças de estado físico. A plataforma oferece recursos versáteis que permitem a criação de ambientes virtuais interativos, nos quais os usuários podem explorar e experimentar de maneira dinâmica.

Durante esta exploração, serão destacadas as funcionalidades específicas do Roblox Studio que facilitam a representação e a interação com os conceitos relacionados às mudanças de estado físico. Vamos mergulhar nessa jornada pelo Roblox Studio e descobrir como ele pode ser uma ferramenta eficaz na construção de Objetos Virtuais de Aprendizagem.

O primeiro passo na construção desta experiência é compreender como e onde o jogador aparecerá no mapa. O local de *spawn* do jogador é denominado ***spawnlocation***. Este local pode ser localizado na aba do **Explorador** como um filho, ou seja, um objeto que está aninhado dentro do **Workspace**. Para acessá-lo, clique na seta no lado esquerdo do **Workspace** e procure por ***spawnlocation***, como mostrado na Figura 28..

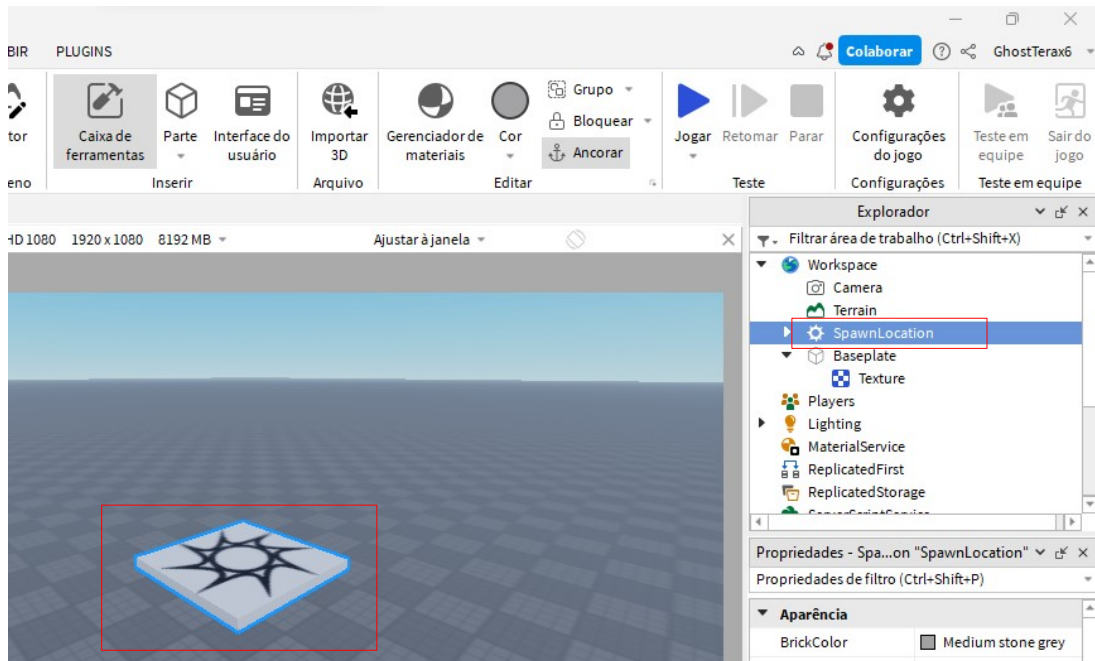
Na Figura 28, é possível visualizar o local de aparecimento do jogador tanto no **Explorador** quanto no ambiente de desenvolvimento, ambos destacados com o retângulo vermelho. Isso serve como uma referência visual para garantir que o ponto de *spawn* esteja configurado conforme desejado. Este é um passo fundamental para definir o ponto inicial da experiência do jogador no ambiente virtual que está sendo construído.

Após compreender onde o jogador aparecerá, você pode iniciar a construção da sua experiência, delimitando a área a ser utilizada por ele. Isso pode ser realizado inserindo objetos, como paredes. Para adicionar uma parede, acesse o menu **Exibir** e certifique-se de que a caixa de ferramentas está selecionada, conforme mostrado na Figura 29.

Em seguida, pesquise o nome do objeto desejado e, ao encontrá-lo, arraste-o para a área da experiência. Alternativamente, você pode inseri-lo seguindo as instruções descritas na seção 3.3. Essas paredes ajudarão a definir os limites do espaço disponível para o jogador, proporcionando uma estrutura clara para a experiência. Certifique-se de posicionar e dimensionar esses objetos de acordo com a visão geral do design que você deseja para a

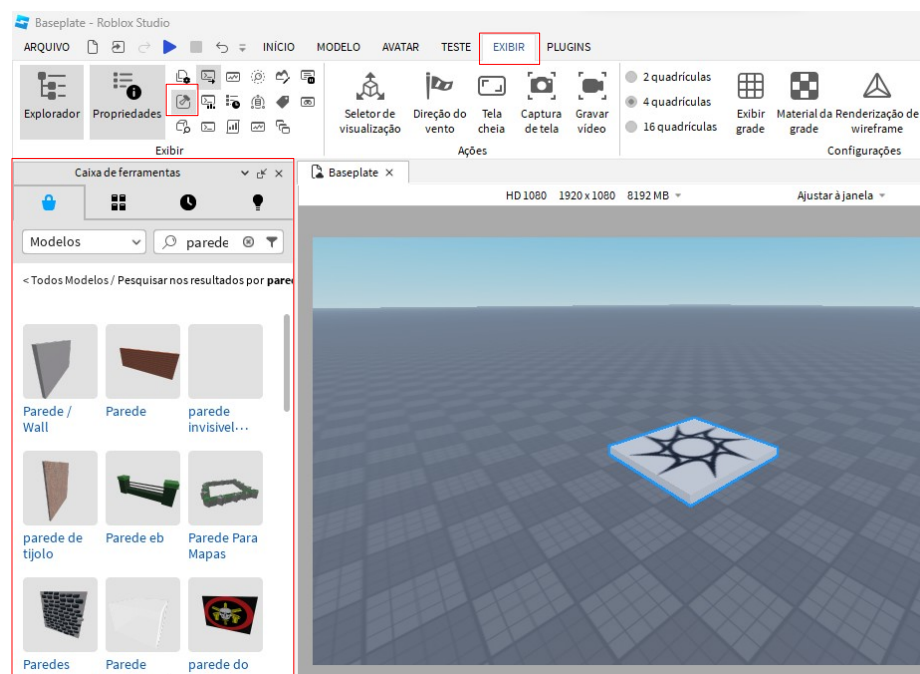
interação do jogador. Essa etapa é crucial para criar um ambiente controlado e envolvente.

Figura 28 - Local de Aparecimento do Jogador



Fonte: O autor (2023).

Figura 29 - Inserção de objetos via Caixa de Ferramentas

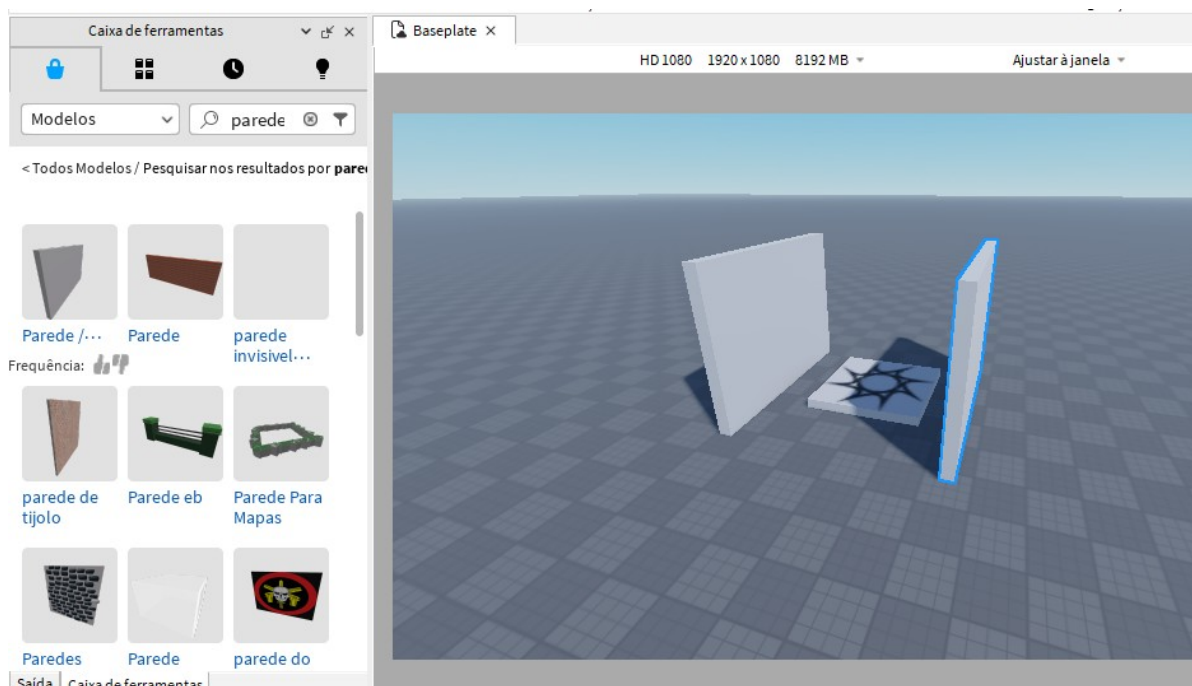


Fonte: O autor (2023).

Ao arrastar os Objetos para a cena, você terá uma visão parecida com a demonstrada na Figura 30. Você tem a liberdade para trabalhar o cenário da forma que preferir, não se

esqueça apenas de ancorar os objetos, para isso, você deve selecionar o objeto e no menu **Início** selecionar a opção **ancorar**, isso faz com que os objetos permaneçam no local que você delimitou.

Figura 30 - Inserção de objetos via Caixa de Ferramentas

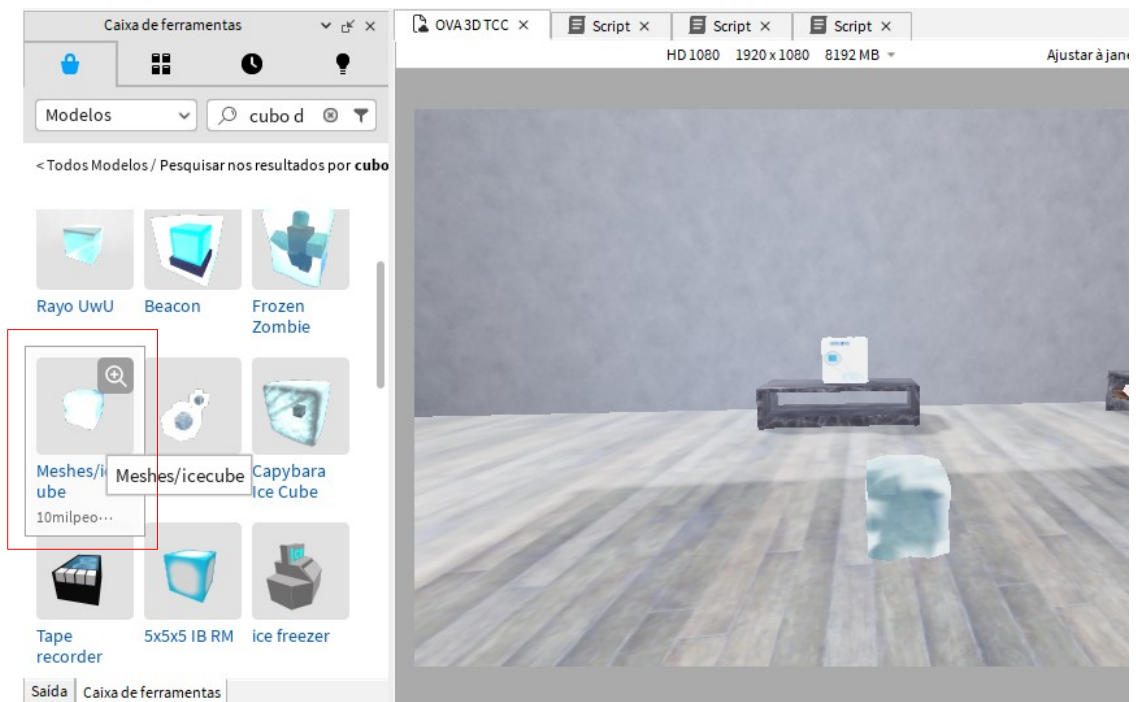


Fonte: O autor (2023).

Após a concepção do cenário desejado, é necessário incorporar as ações que o personagem executará ao interagir com objetos específicos. Recomenda-se a construção de um ambiente semelhante à representação visualizada na Figura 31, o qual servirá como base para o desenvolvimento das demais funcionalidades abordadas neste tutorial. Esse cenário é constituído de quatro paredes estruturais, ancoradas para delimitar um espaço retangular, além de um teto e um piso. Os elementos do teto e do piso podem ser os mesmos utilizados nas paredes, bastando uma rotação de 90° e uma alocação que complemente o espaço construído.

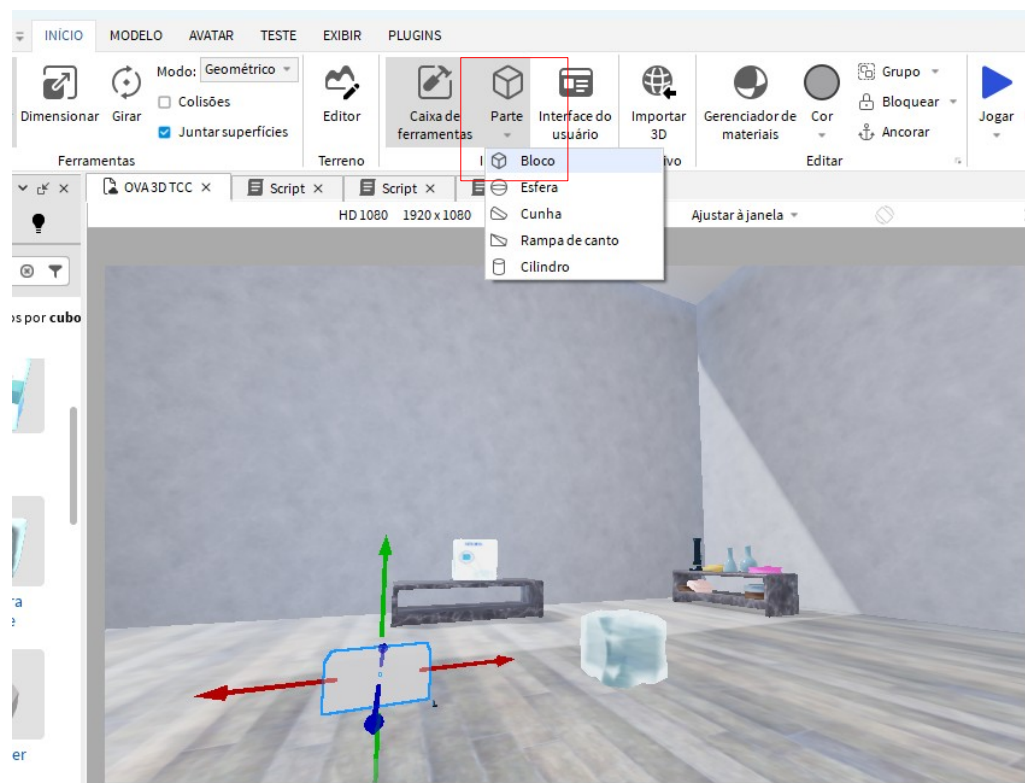
Além disso, o cenário possui três mesas, e algumas estantes, de forma que se assemelhe a um laboratório de química, esses objetos podem ser encontrados da mesma forma que as paredes, bastando buscá-los na “Caixa de Ferramentas”. Você possui liberdade criativa para construir seu cenário.

Figura 32 - Inserção de cubo de gelo



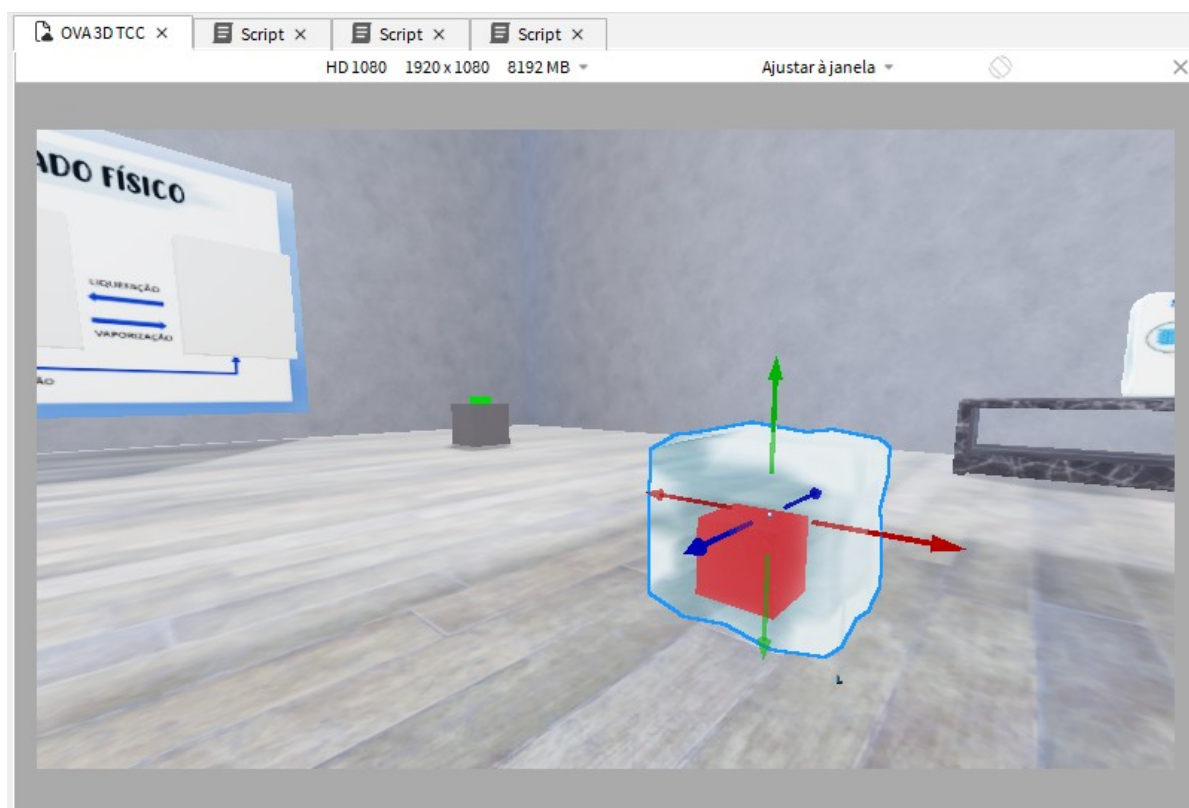
Fonte: O autor (2023).

Figura 33 - Inserção de retângulo



Fonte: O autor (2023).

Figura 34 - Inserção de retângulo

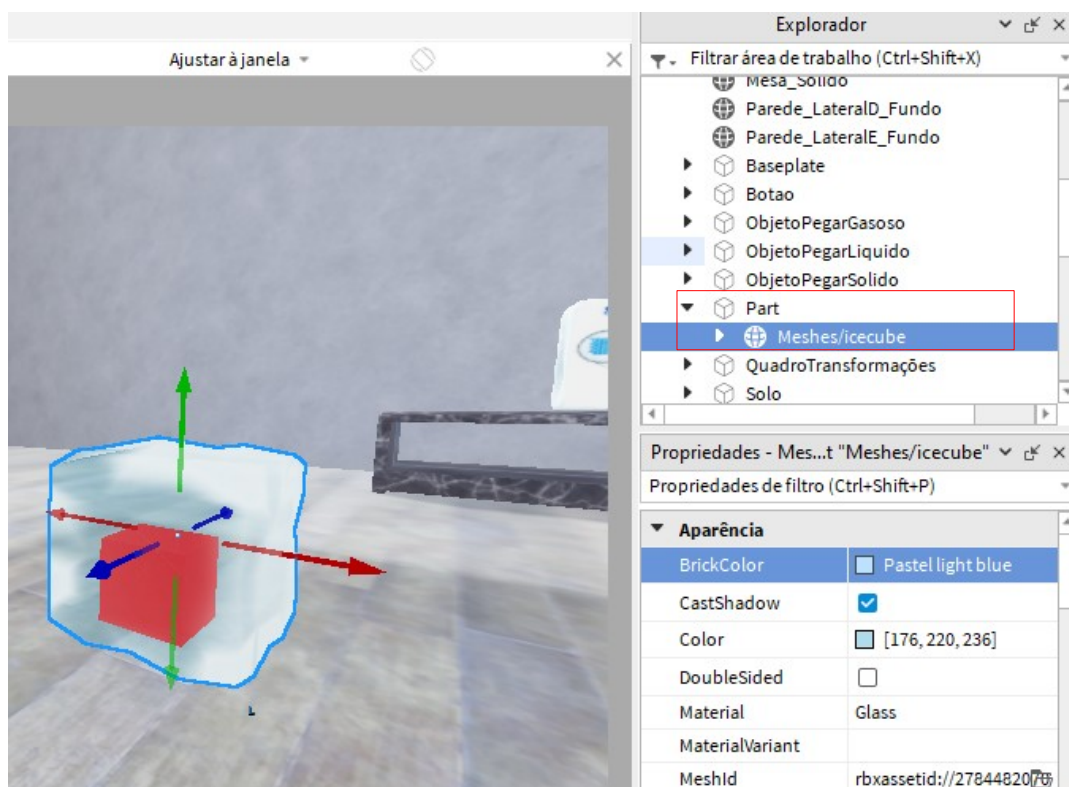


Fonte: O autor (2023).

Após posicionar visualmente o Cubo de Gelo sobre o **Bloco** na janela de cenário, localize os componentes denominados **Meshes/icecube** e **Part** na janela **Explorador**. Você pode identificar esses componentes clicando sobre seus nomes; eles serão destacados com um contorno azul no cenário, como exemplificado na Figura 34 do cubo de gelo. Ao identificar esses itens, arraste o componente **Meshes/icecube** e posicione-o sobre a **Part**, de modo que o primeiro se torne um subcomponente do segundo na hierarquia do **Explorador**, conforme ilustrado na Figura 35. Ao realizar esse procedimento, você cria um objeto único, onde o cubo de gelo funciona como uma espécie de revestimento para o **Bloco**, permitindo interações com o jogador.

Caso deseje mudar o nome do objeto **part**, você pode clicar sobre ele com o botão direito do mouse, e clicar em **Renomear**, em seguida digitar o nome que desejar, uma sugestão é utilizar **Cubo de Gelo**.

Figura 35 - Mudança de Hierarquia de objetos



Fonte: O autor (2023).

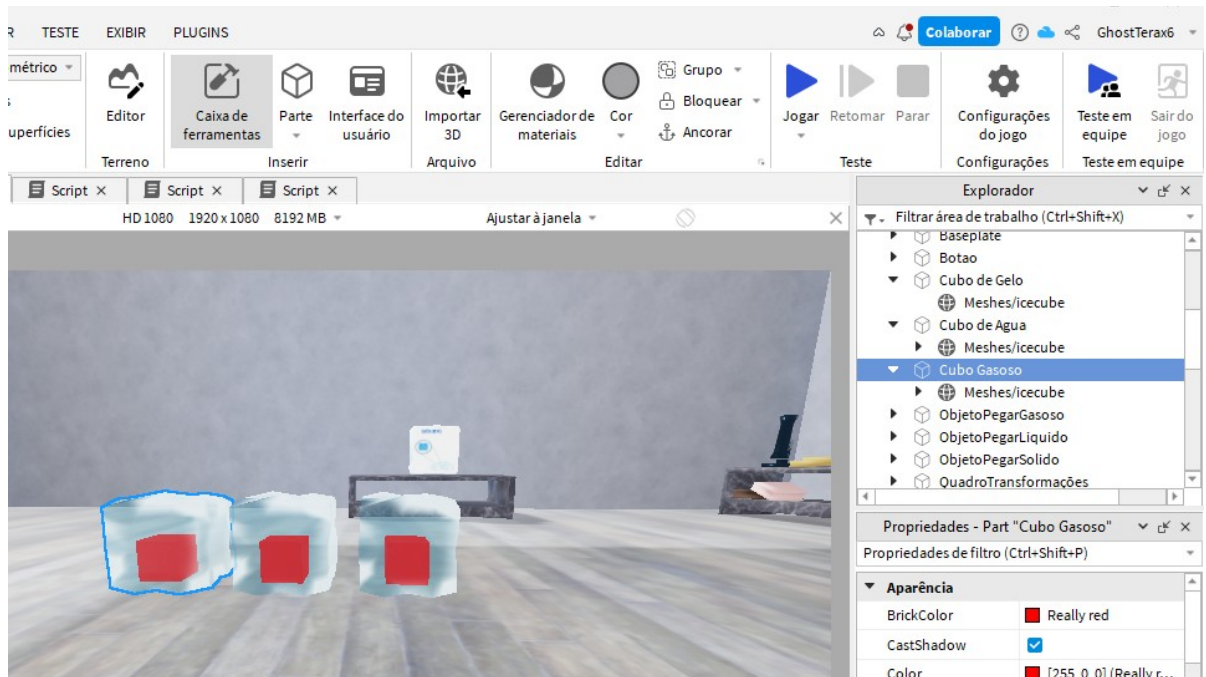
Para evitar retrabalho, você pode duplicar o objeto em questão, basta novamente clicar com o botão direito do mouse sobre ele e clicar em **Duplicar**, ou apertar o atalho **ctrl+d**. Faça isso, de forma que você tenha como resultado final, três cubos de gelo. Assim como demonstra Figura 36. Você pode renomear os cubos como preferir, uma sugestão é chamá-los de **Cubo de Gelo**, **Cubo de Água** e **Cubo de Gás**.

O próximo passo para a construção da experiência é colocar na superfície dos cubos o que o Roblox chama de **Decal**, uma imagem. Nesse ponto, você já deve ter compreendido o porquê de serem criados três cubos, um para cada estado da matéria. Essa imagem pode ser colocada da mesma forma que as paredes da experiência, você deve buscar na **Caixa de Ferramentas**, imagens, para poder posicioná-las sobre as faces dos cubos. Isso pode ser feito, alterando-se o dropdown para imagens e buscando por **water**, sugere-se buscar em inglês os componentes por haver maior abundância de componentes com nomes nesse idioma. Assim como ilustra a Figura 37.

Ao fazer isso, utilize a barra de rolagem e escolha a imagem que melhor se identifica, com sugestão procure por uma chamada **Water Light Pattern**, ao encontrá-la, arraste-a sobre

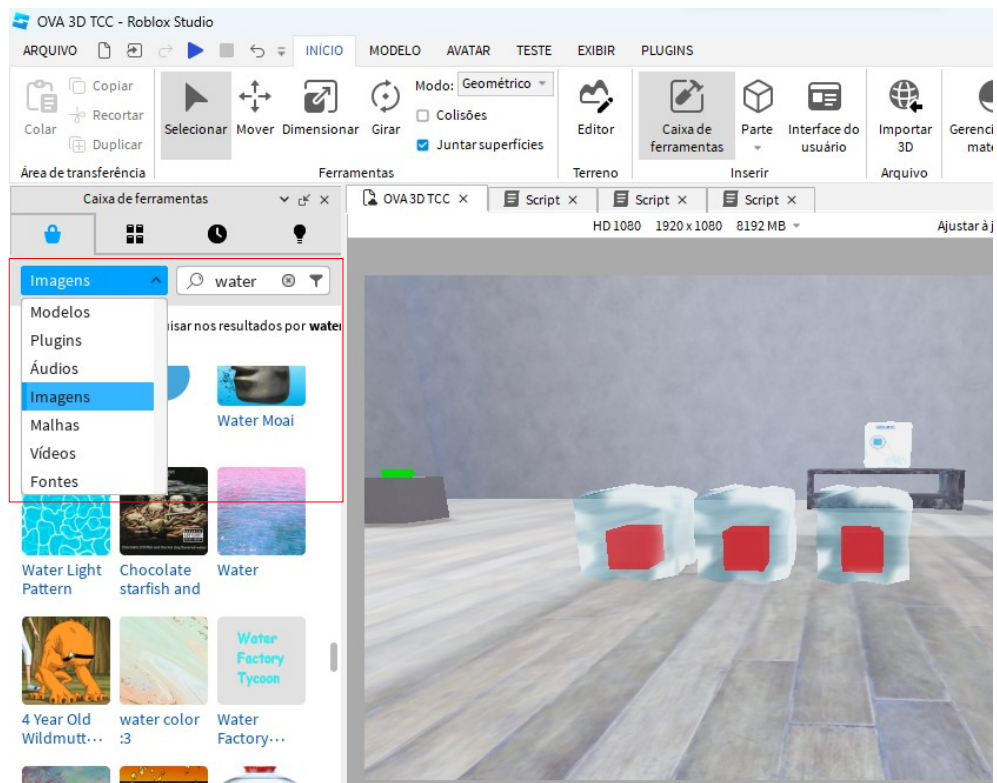
a superfície do Cubo.

Figura 36 - Cubos



Fonte: O autor (2023).

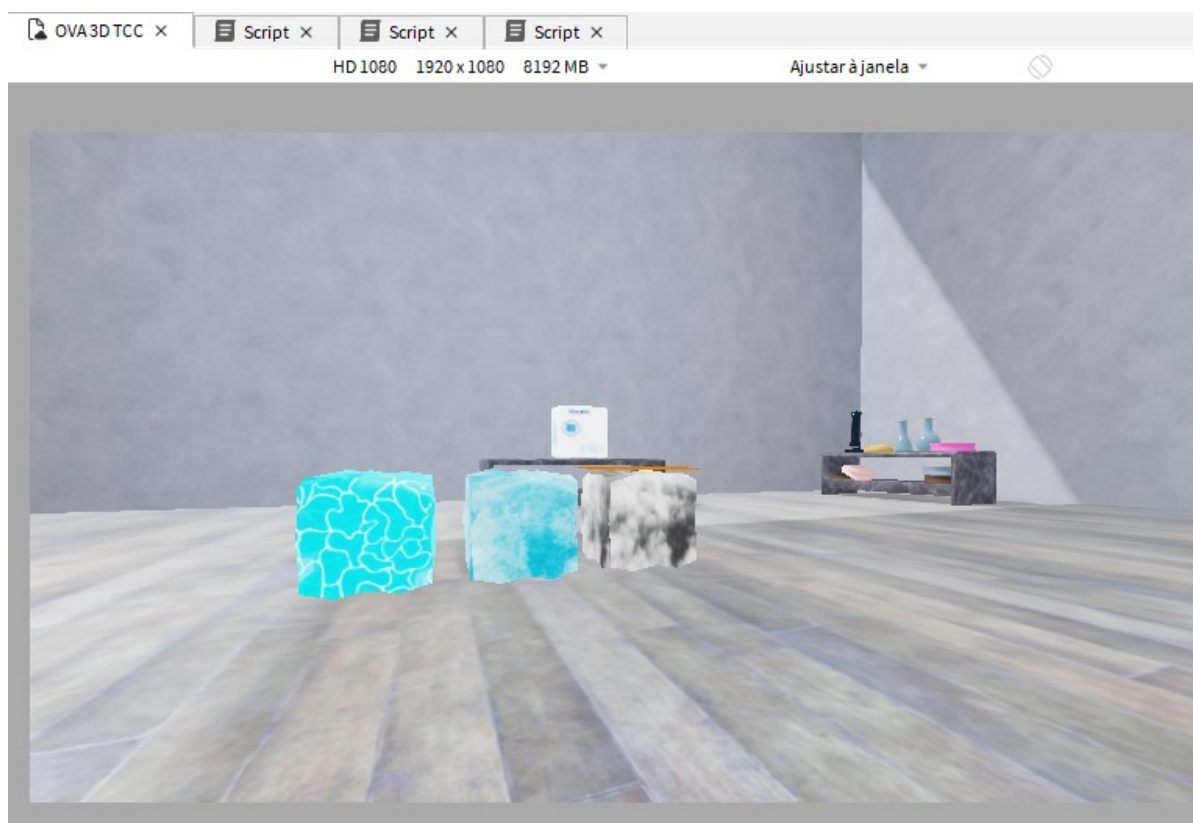
Figura 37 - Imagens na Caixa de Ferramentas



Fonte: O autor (2023).

Posicione imagens em todas as faces do cubo, de forma que ele pareça ser inteiramente desse material, assim como demonstra a Figura 38. Faça isso, para todos os cubos.

Figura 38 - Cubos com Texturas

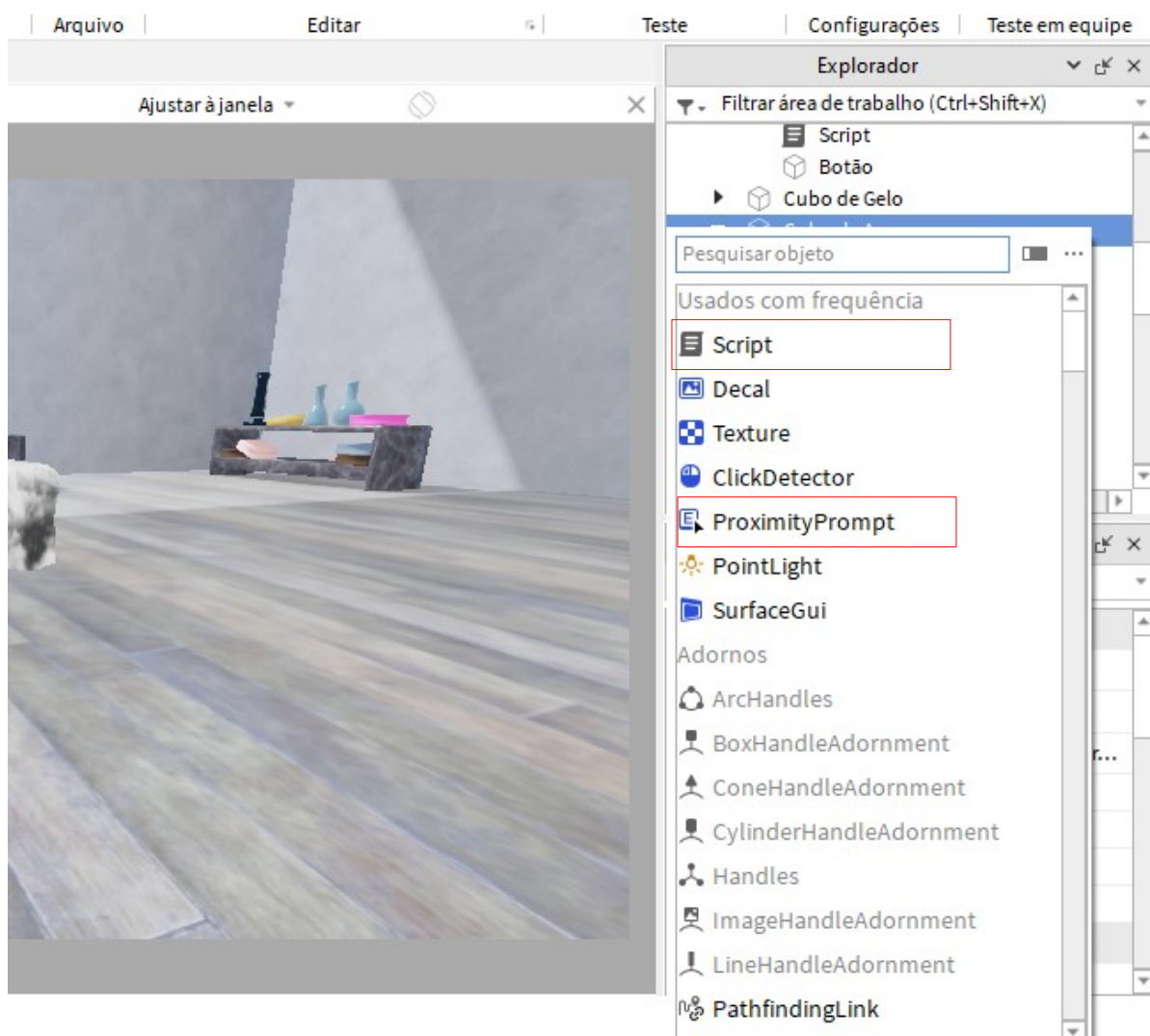


Fonte: O autor (2023).

Nesse ponto, os cubos estão visualmente prontos, basta agora que o comportamento deles seja definido, isso será feito por meio de *Scripts* e componentes auxiliares. Identifique na hierarquia, os cubos criados, usaremos como exemplo o cubo de água. Ao encontrá-lo, posicione o mouse sobre ele e acrescente um objeto chamado **ProximityPrompt**, clicando no + que aparece do lado direito do objeto, acrescente também o objeto *script*. A Figura 38 evidencia esses objetos.

O **ProximityPrompt**, trata-se de um objeto que permite identificar a aproximação do jogador com relação ao objeto que o possui como filho, sendo assim, ele permite uma interação via teclado de jogadores que estão a uma certa distância dele.

Figura 39 - Inserir Objetos Script e ProximityPrompt



Fonte: O autor (2023).

Após inserir os objetos, dê dois cliques sobre o objeto **Script**. Isso fará com que o espaço para escrita de códigos em Lua apareça. Nessa aba, você encontrará um trecho **print("Hello world!")**, que exibirá **Hello world!** no console do Roblox Studio. Substitua esse trecho pelo código a seguir.

As três primeiras linhas desse código, dizem respeito à configuração das variáveis locais, as quais serão utilizadas para se referir aos objetos no contexto do **Script**. Pode-se dizer que na primeira linha, temos a seguinte interpretação: a variável local **Prompt_proximidade**, pode ser encontrada acessando-se o pai do **script**, ou seja, o objeto ao qual o **script** foi inserido. O mesmo se repete para as linhas seguintes, em que é atribuído

como o pai da variável o pai do **script** e identifica-se a variável **plr**, como o jogador local, que pode ser encontrado no contexto global do game, dentre os *players* disponíveis.

Em lua, assim como várias outras linguagens, o **.**, indica o acesso a propriedades de uma variável ou classe, sendo assim, ao utilizar **Prompt_proximidade.ActionText**, é o mesmo que dizer que você está alterando o texto de ação quando o jogador se aproxima de um objeto, ou seja, o texto que será exibido.

De maneira geral, o Script 1 faz com que seja mostrado **Pegar Líquido** quando o jogador se aproxima do cubo de água. Após se aproximar, o jogador terá a possibilidade de interagir com o objeto por meio da Tecla E que é definida por padrão no **ProximityPrompt**. Ao pressionar essa tecla por 1 segundo ele equipará o item e fará com que o bloco desapareça.

Script 1 – Script para Coletar o Item Liquido

```

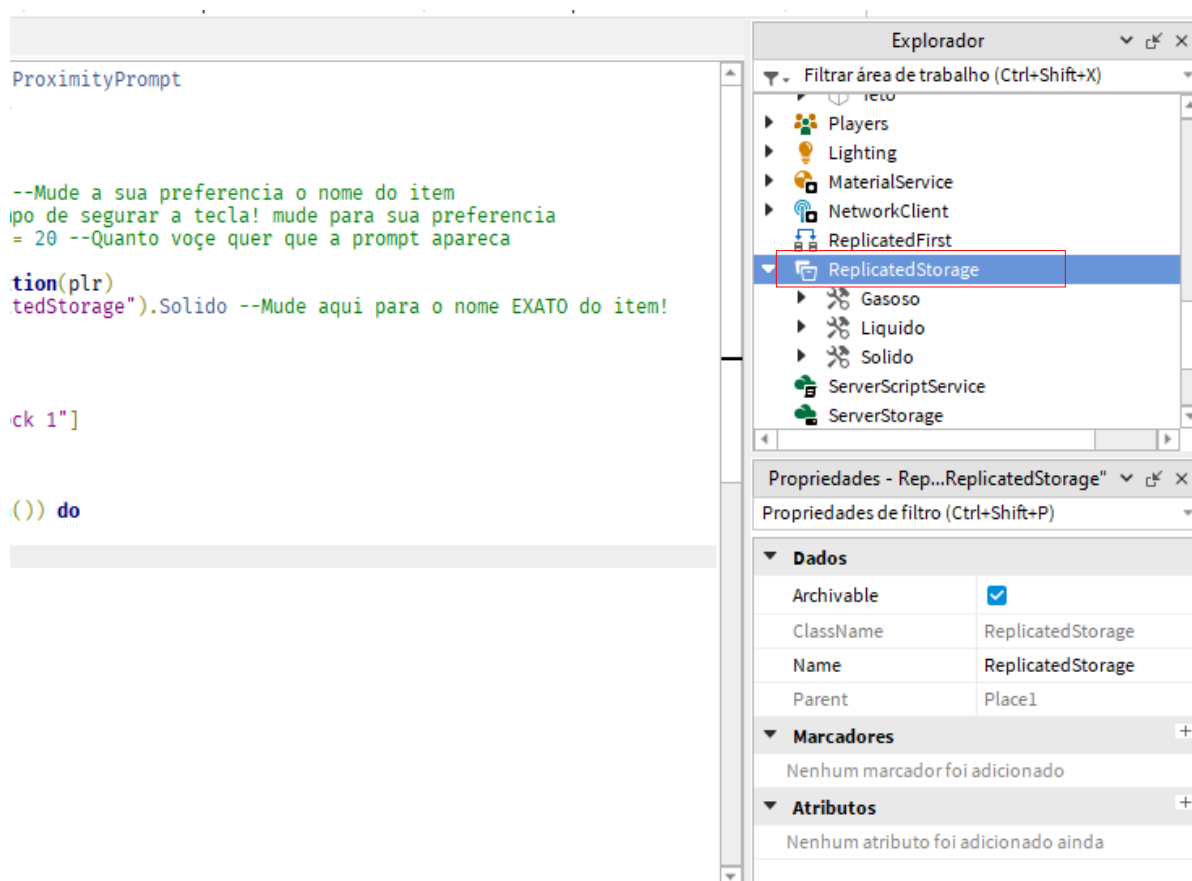
1. local Prompt_proximidade = script.Parent.ProximityPrompt
2. Prompt_proximidade.Parent = script.Parent
3. local plr = game.Players.LocalPlayer
4. --CONFIGURAÇÕES DO PROMPT
5. Prompt_proximidade.ActionText = "Pegar"
6. Prompt_proximidade.ObjectText = "Liquido" --Mude a sua preferencia o
   nome do item
7.
8. Prompt_proximidade.HoldDuration = 1 --Tempo de segurar a tecla! mude
   para sua preferencia
9. Prompt_proximidade.MaxActivationDistance = 20 --Quanto voce quer que
   a prompt apareca
10.
11.     Prompt_proximidade.Triggered:Connect(function(plr)
12.         local Item = game.GetService("ReplicatedStorage").Liquido
13.         --Mude aqui para o nome EXATO do item!
14.         local Clone = Item:Clone()
15.         Clone.Parent = plr.Character
16.         local pai = script.Parent["Meshes/Block 1"]
17.         pai.Transparency = 1
18.         script.Parent.Transparency = 1
19.         for _, filho in pairs(pai:GetChildren()) do
20.             if filho:IsA("Decal") then
21.                 filho.Transparency = 1
22.             end
23.         end
24.         Prompt_proximidade.Enabled = false
25.     end)

```

Fonte: O Autor (2023)

Para que este código funcione adequadamente, você deve adicionar no **ReplicatedStorage** na aba do “Explorador”, um objeto do tipo **Tool**, o qual pode conter uma **Mesh**, essa última se trata da aparência do item. A Figura 40, demonstra onde encontra-se o **ReplicatedStorage**. Para adicionar a **Tool** basta seguir o mesmo procedimento previamente explicitado, onde deve-se posicionar o mouse sobre o objeto e clicar em +, após isso buscar por **Tool** na barra de pesquisas.

Figura 40 - Localizar ReplicatedStorage



Fonte: O autor (2023).

Acrescente um **Tool** para cada cubo, e renomeie-os como Líquido, Gasoso e Solido. Após realizar esses passos, repita o processo para os outros dois cubos, acrescentando um **script** e um **ProximityPrompt** par cada, e teste a experiência.

O próximo passo, é criar um local para depositar os objetos que serão pegos pelos jogadores, nesse caso um quadro com as transformações da matéria. Você tem liberdade para

criar o plano de fundo da forma que preferir, uma vez que ele é meramente ilustrativo, o mais importante são os locais onde o jogador interagirá.

Você pode seguir o mesmo modelo utilizado para criar os cubos, onde três *parts* devem ser inseridas no contexto da experiência, e nelas devem ser inseridos por meio do “+” na aba do **Explorador**, um *script* e um **ProximityPrompt**. Elas podem ser posicionadas assim como demonstra a Figura 41.

Figura 41 - Quadro de Transformações



Fonte: O autor (2023).

Para o quadro de transformações, você deve adicionar em cada componente *script* o código do Script 2. Lembre-se de alterar os nomes dos componentes de acordo com os nomes que você definiu, pois assim você evita que ocorram erros e a experiência funcione de forma adequada.

Script 2 – Script para Coletar o Item Líquido

```

1. local Prompt_proximidade = Instance.new("ProximityPrompt")
2. Prompt_proximidade.Parent = script.Parent
3. local plr = game.Players.LocalPlayer
4. local obj = script.Parent
5. --CONFIGURAÇÕES DO PROMPT
6. Prompt_proximidade.ActionText = "Pegar"
7. Prompt_proximidade.ObjectText = "Item" --Mude a sua preferencia o nome do item
8. Prompt_proximidade.HoldDuration = 1 --Tempo de segurar a tecla! mude para sua preferencia
9. Prompt_proximidade.MaxActivationDistance = 20 --Quanto voce quer que a prompt apareca
  
```

```

10.
11. Prompt_proximidade.Triggered:Connect(function(plr)
12.     local itemEquipado = plr.Character:FindFirstChildWhichIsA("Tool")
13.     if(itemEquipado)then
14.         local clone = itemEquipado:Clone()
15.         clone.Parent = plr.Backpack
16.         itemEquipado:Remove()
17.         local itemMochila = plr.Backpack:FindFirstChildWhichIsA("Tool")
18.
19.         if(obj.Decal.Texture == "http://www.roblox.com/asset/?id=15558862920") then
20.             if(itemMochila.Name=="Solido")then
21.                 obj.Decal.Texture = "rbxassetid://15177194321"
22.                 plr.Backpack:ClearAllChildren()
23.             elseif(itemMochila.Name == "Liquido")then
24.                 obj.Decal.Texture = "rbxassetid://15177207526"
25.                 plr.Backpack:ClearAllChildren()
26.             elseif(itemMochila.Name==" Gasoso")then
27.                 obj.Decal.Texture = "rbxassetid://15177208468"
28.                 plr.Backpack:ClearAllChildren()
29.             end
30.         else
31.             local screenGui = Instance.new("ScreenGui")
32.             screenGui.Parent = plr:FindFirstChild("PlayerGui") or
plr:WaitForChild("Backpack")
33.             clone.Parent = plr.Character
34.             -- Crie um objeto "TextLabel" com a mensagem
35.             local mensagem = Instance.new("TextLabel")
36.             mensagem.Size = UDim2.new(0, 800, 0, 50) -- Tamanho da mensagem
37.             mensagem.Position = UDim2.new(0.5, -100, 0.5, -25) -- Posição central
38.             mensagem.BackgroundColor3 = Color3.new(0, 0, 0) -- Cor de fundo
39.             mensagem.TextColor3 = Color3.new(1, 1, 1) -- Cor do texto
40.             mensagem.FontSize = Enum.FontSize.Size24
41.             mensagem.Text = "Já existe um Objeto aqui, posicione em outro local" --
    Texto da mensagem
42.             mensagem.Parent = screenGui
43.
44.             -- Remova a mensagem após um tempo (opcional)
45.             wait(5) -- Espera por 5 segundos
46.             mensagem:Remove() -- Remove a mensagem
47.         end
48.     end
49. end)
1.

```

Fonte: O Autor (2023)

Agora, basta criar um botão para reiniciar a experiência, o design dele ficará por sua conta, um treinamento para reforçar os conceitos demonstrados nesse tutorial. O Script 3 demonstra o código que define o comportamento do botão.

Script 3 – Script Reiniciar a experiência

```

1. local Prompt_proximidade = Instance.new("ProximityPrompt")
2. Prompt_proximidade.Parent = script.Parent
3. local plr = game.Players.LocalPlayer
4. --CONFIGURAÇÕES DO PROMPT
5. Prompt_proximidade.ActionText = "Pegar"
6. Prompt_proximidade.ObjectText = "Item" --Mude a sua preferencia o nome do item
7. Prompt_proximidade.HoldDuration = 1 --Tempo de segurar a tecla! mude para sua preferencia
8. Prompt_proximidade.MaxActivationDistance = 20 --Quanto voce quer que a prompt apareca
9. Prompt_proximidade.Triggered:Connect(function(plr)

10.     local gasoso = script.Parent.Parent:FindFirstChild("SoltarGasoso")
11.     local liquido = script.Parent.Parent:FindFirstChild("SoltarLiquido")
12.     local solido = script.Parent.Parent:FindFirstChild("SoltarSolido")
13.     local pegarSolido = script.Parent.Parent:FindFirstChild("ObjetoPegarSolido")
14.     local pegarLiquido = script.Parent.Parent:FindFirstChild("ObjetoPegarLiquido")
15.     local pegarGasoso = script.Parent.Parent:FindFirstChild("ObjetoPegarGasoso")
16.
17.     if(gasoso.Decal.Texture ~= "http://www.roblox.com/asset/?id=15558862920" and
liquido.Decal.Texture ~= "http://www.roblox.com/asset/?id=15558862920" and
solido.Decal.Texture ~= "http://www.roblox.com/asset/?id=15558862920") then
18.
19.         if(gasoso.Decal.Texture == "rbxassetid://15177208468")then
20.             gasoso.Decal.Color3 = Color3.new(0, 1, 0)
21.         else
22.             gasoso.Decal.Color3 = Color3.new(1, 0, 0)
23.         end
24.
25.         if(liquido.Decal.Texture == "rbxassetid://15177207526")then
26.             liquido.Decal.Color3 = Color3.new(0, 1, 0)
27.         else
28.             liquido.Decal.Color3 = Color3.new(1, 0, 0)
29.         end
30.
31.         if(solido.Decal.Texture == "rbxassetid://15177194321")then
32.             solido.Decal.Color3 = Color3.new(0, 1, 0)
33.         else
34.             solido.Decal.Color3 = Color3.new(1, 0, 0)
35.         end
36.

```

```

37.      wait(3)
38.      pegarGasoso.ProximityPrompt.Enabled = true
39.      pegarSolido.ProximityPrompt.Enabled = true
40.      pegarLiquido.ProximityPrompt.Enabled = true
41.
42.
43.      local pais = {}
44.      table.insert(pais, pegarGasoso["Meshes/Block 1"])
45.      table.insert(pais, pegarSolido["Meshes/Block 1"])
46.      table.insert(pais, pegarLiquido["Meshes/Block 1"])
47.
48.
49.      script.Parent.Transparency = 0
50.      for _, pai in pairs(pais) do
51.          pai.Transparency = 0
52.          for _, filho in pairs(pai:GetChildren()) do
53.              if filho:IsA("Decal") then
54.                  filho.Transparency = 0
55.              end
56.          end
57.      end
58.
59.      gasoso.Decal.Texture = ""
60.      liquido.Decal.Texture = ""
61.      solido.Decal.Texture = ""
62.      solido.Decal.Color3 = Color3.new(0.898039, 0.898039, 0.898039)
63.      liquido.Decal.Color3= Color3.new(0.898039, 0.898039, 0.898039)
64.      gasoso.Decal.Color3= Color3.new(0.898039, 0.898039, 0.898039)
65.  end
66. end)
67.

```

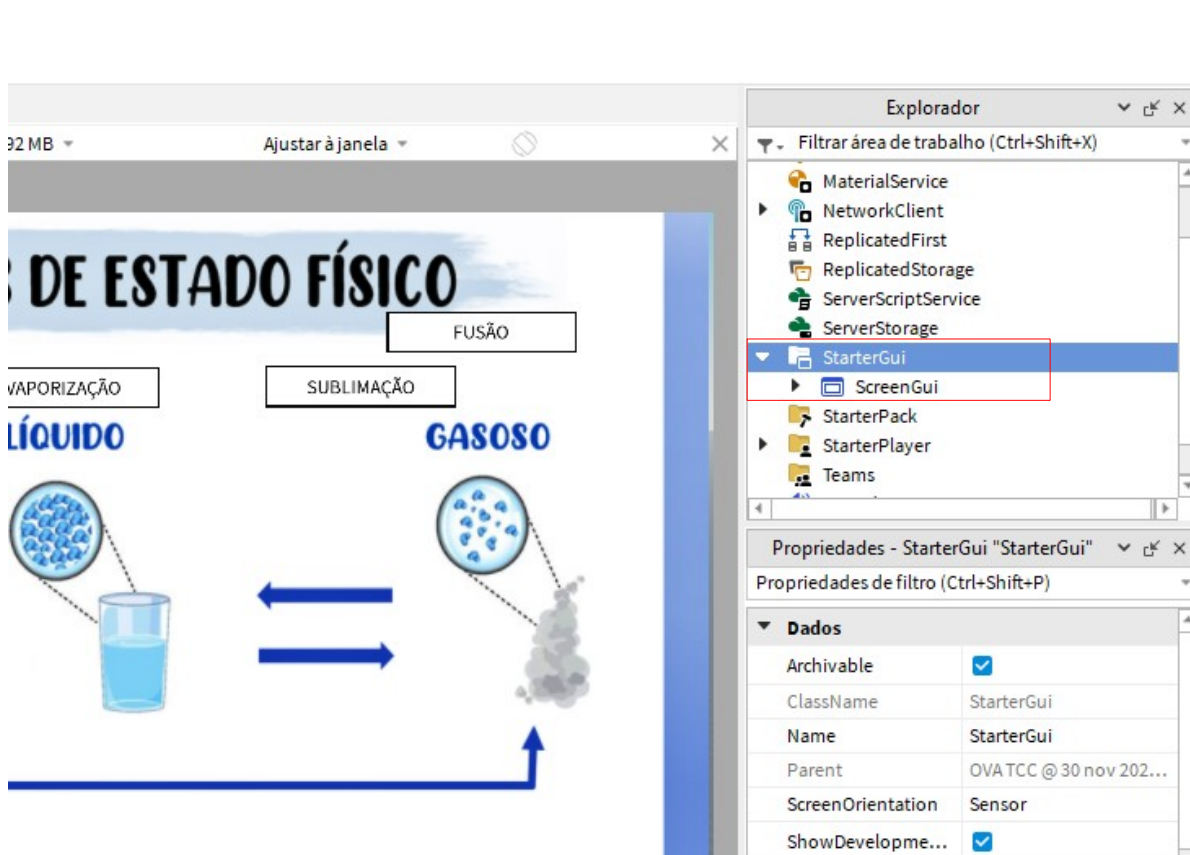
Fonte: O Autor (2023)

Agora basta testar sua experiência e adaptá-la da forma mais adequada às suas necessidades. A experiência evidenciada aborda a maioria das funcionalidades utilizadas, sendo assim você possui agora uma base e um exemplo para realizar suas próprias construções 3D.

6 Construção de uma Experiência 2D

Para construir uma experiência 2D, o primeiro passo é localizar na aba **Explorador** o campo **StarterGui**. Esse campo é responsável por exibir na tela os componentes da Gui Interface, ou seja, os componentes 2D do Roblox. Após encontrar esse componente, inclua uma **ScreenGui**, assim como demonstra a Figura 42.

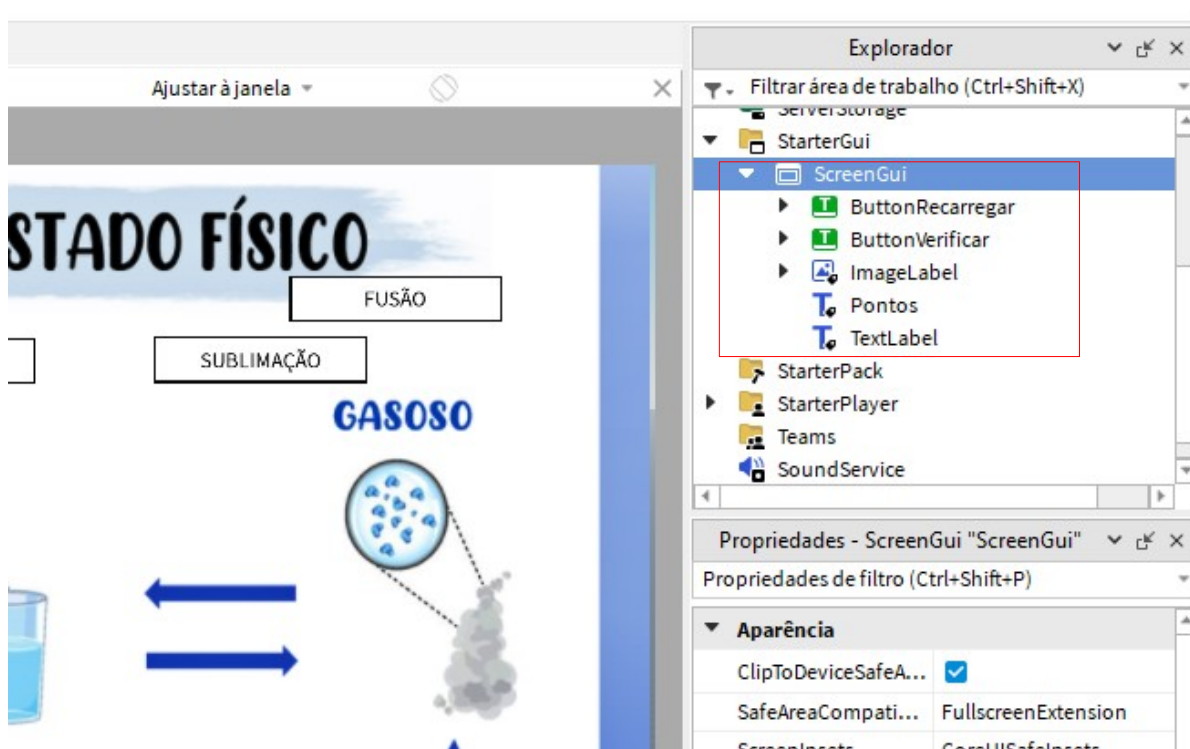
Figura 42 - Incluir ScreenGui



Fonte: O autor (2023).

Após inserir o **ScreenGui**, adicione dentro dele, dois “TextButton”, duas “TextLabel” e uma **ImageLabel**. Você pode renomeá-los como achar necessário, porém segue um exemplo na Figura 43. Deixe sua estrutura exatamente como a demonstrada, para que os *scripts* funcionem de forma adequada.

Figura 43 - Estrutura da Interface

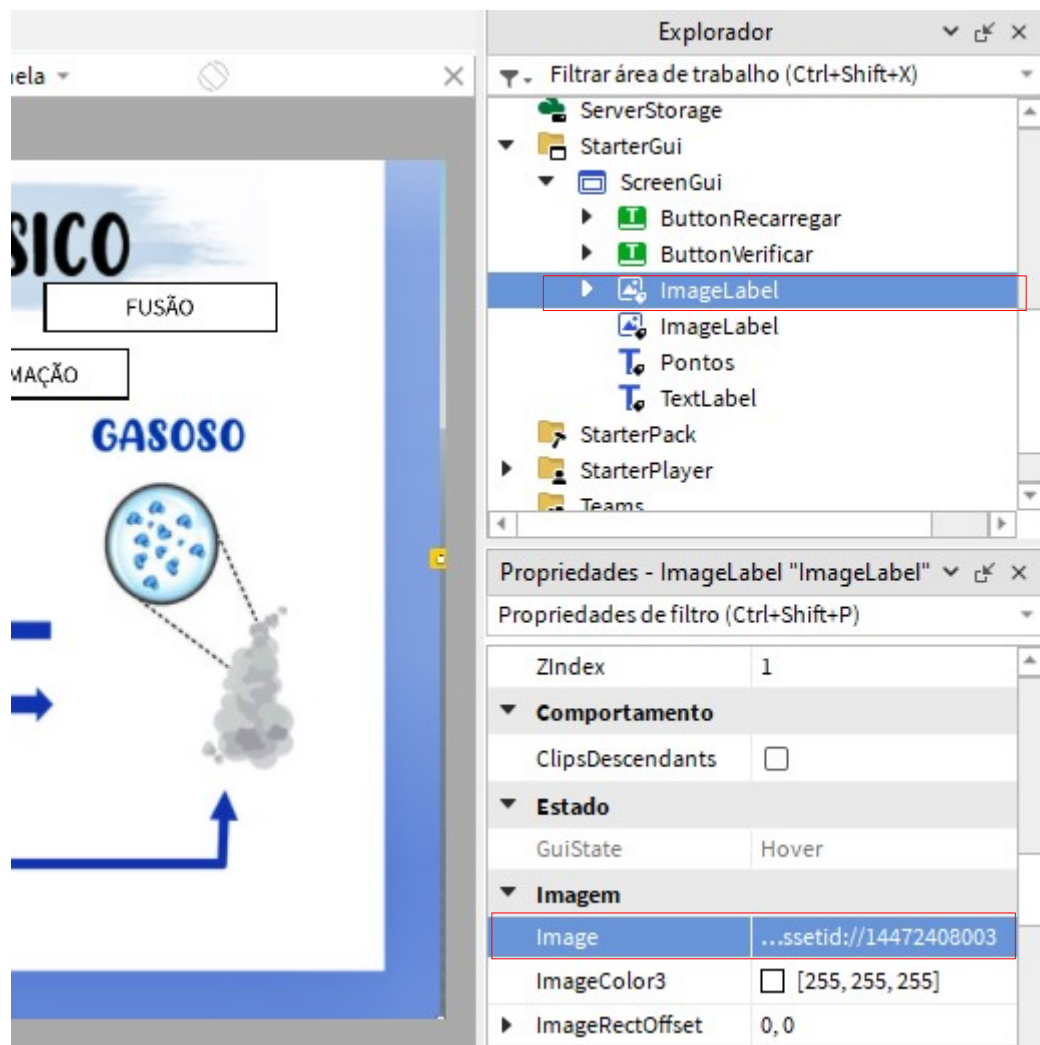


Fonte: O autor (2023).

Após inserir todos os componentes, citados anteriormente, localize sua **ImageLabel** e o campo **Imagem** da mesma, esse localizado na janela propriedades. Lembre-se de conferir se essa janela está habilitada, assim como demonstrado na Figura 21. Ao fazer isso, acesse o link [Images/ova h5p - Mercado de Criações \(roblox.com\)](https://www.roblox.com/asset/?id=14472408003). Esse link possui a imagem utilizada por padrão nas experiências desse tutorial, clique em **Tentar no Studio**, e ela será adicionada à sua coleção. Após isso, clique no campo “Imagem”, assim como demonstrado na Figura 44. Para acessar esse campo primeiro você deve clicar sobre a **ImageLabel** e depois encontrá-lo na janela **Propriedades**.

Após adicionar a imagem de plano de fundo na sua coleção, cole a seguinte URL: `rbxassetid://14472408003`, na janela exibida ao clicar na propriedade “Imagem”. Isso fará com que o plano de fundo do **ImageLabel** seja definido como o padrão utilizado nas experiências desse tutorial, assim como demonstrado na Figura 45. Após isso, redimensione-o para ocupar toda a tela do dispositivo.

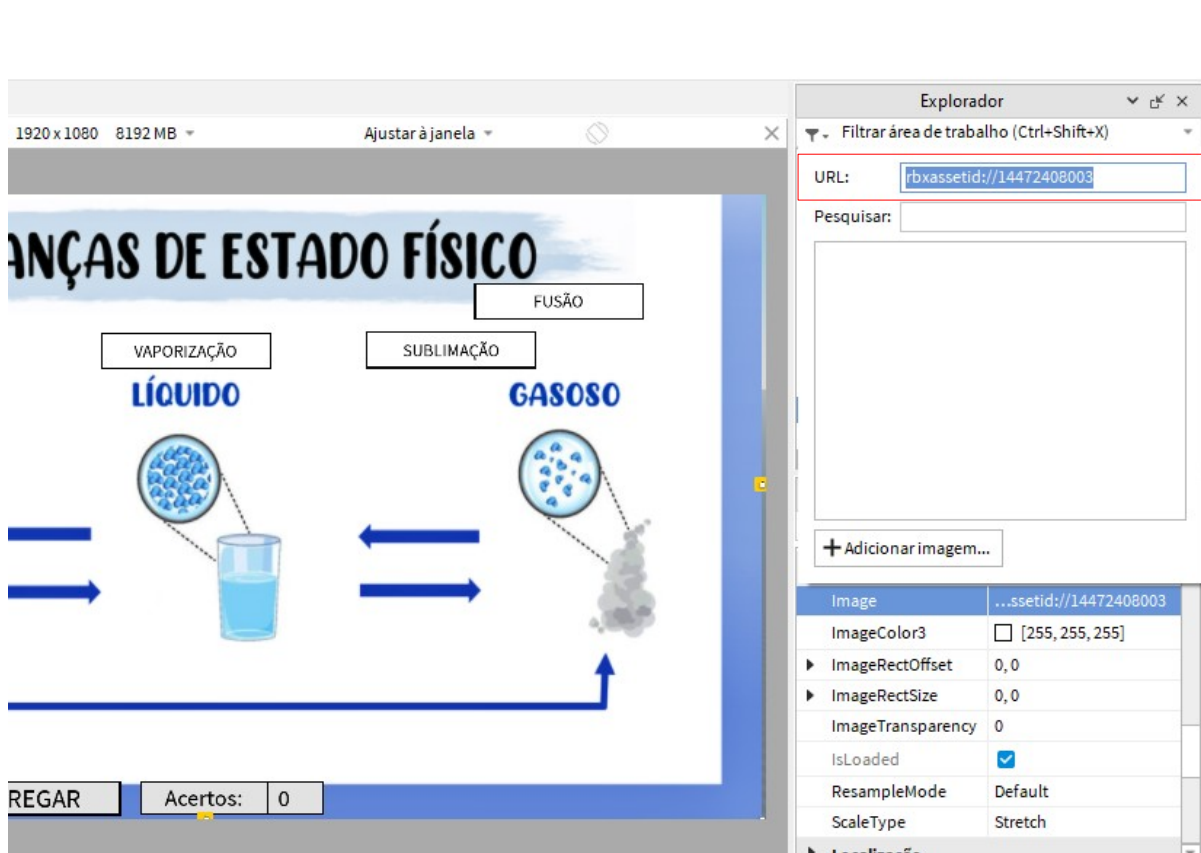
Figura 44 - Estrutura da Interface



Fonte: O autor (2023).

Certifique-se de que o plano de fundo foi inserido adequadamente, isso pode ser verificado visualmente, em que a imagem para plano de fundo substituirá a imagem padrão do componente **ImageLabel**.

Figura 45 - Plano de Fundo ImageLabel

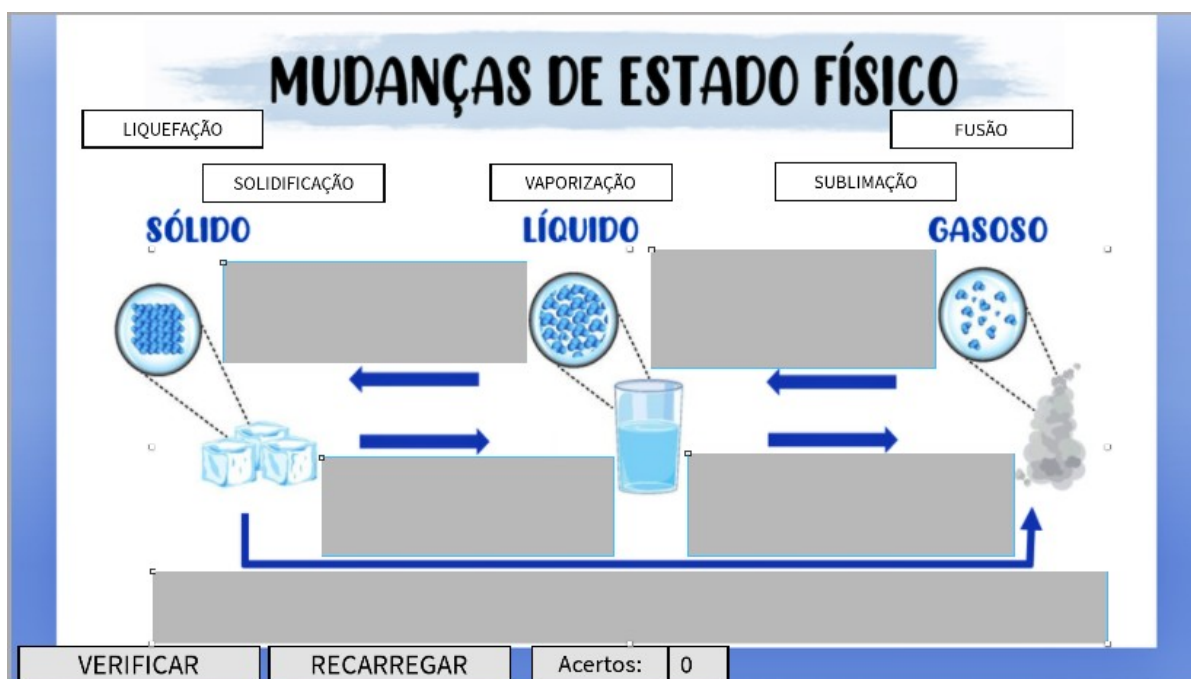


Fonte: O autor (2023).

Após redimensionar a **ImageLabel** para ocupar toda a tela, faça com que o layout da experiência que está construída fique o mais próximo possível daquele demonstrado na Figura 46. Para isso, insira como um filho da **ImageLabel**, cinco **TextLabel** e cinco **Frames**, na figura eles estão destacados na cor cinza, porém por padrão eles não terão essa cor, não é necessário que você a altere, isso foi feito apenas para facilitar a visualização de onde serão posicionados.

Para alterar o texto das **Label's**, você deve clicar sobre elas na janela do **Explorador** e localizar na janela de propriedades o campo **Text**, e assim substituir o conteúdo pelo que você desejar.

Figura 46 - Layout da experiência 2D



Fonte: O autor (2023).

Após isso, inclua em cada componente **Frame**, um objeto **BoolValue** e renomeie como **TemLabel**. Aproveite e inclua também um **Script** para cada **Frame**, um para a **ImageLabel** e também um para cada **TextButton**. Eles serão utilizados para controlar o comportamento da experiência.

Para o *script* de movimentação das **Label's**, aquele com está dentro da **ImageLabel**, você deve utilizar o código evidenciado no Script 4 . Esse *script* faz com que seja possível movimentar uma quantidade variável de **Label's**, sendo assim ele é generalista e faz com que a posição do objeto se altere de acordo com o arraste do mouse.

Script 4 – Script para Coletar o Item Liquido

```

1. -- Referências aos objetos
2. local screenGui = script.Parent -- A ScreenGui que contém as TextLabels
3. local mouse = game.Players.LocalPlayer:GetMouse() -- Objeto mouse
4. local sublimacao = script.Parent.Sublimacao
5. -- Tabela para armazenar informações de arrasto para cada TextLabel
6. local draggingInfo = {}
7. local fusao = script.Parent.Fusao
8. local solidificacao = script.Parent.Solidificacao
9. local liquefacao = script.Parent.Liquefacao

```

```

10. local vaporizacao = script.Parent.Vaporizacao
11.

12. -- Obtém a resolução da tela do jogador
13. local player = game.Players.LocalPlayer
14. local screenWidth = player.PlayerGui.ScreenGui.AbsoluteSize.X
15. local screenHeight = player.PlayerGui.ScreenGui.AbsoluteSize.Y
16.

17. -- Configura o tamanho da ImageLabel para cobrir toda a tela
18. local imageLabel = script.Parent -- Assumindo que o script está dentro da
    ImageLabel
19. imageLabel.Size = UDim2.new(0, screenWidth, 0, screenHeight)
20.

21. -- Funções para manipulação do arrasto
22. local function onMouseButton1Down(label)
23.     local offset = label.Position - UDim2.new(0, mouse.X, 0, mouse.Y)
24.     draggingInfo[label] = {
25.         isDragging = true,
26.         offset = offset
27.     }
28. end
29.

30. local function onMouseButton1Up(label)
31.     draggingInfo[label] = nil
32. end
33.

34. local function onMouseMove()
35.     for label, info in pairs(draggingInfo) do
36.         if info.isDragging then
37.             local newPosition = UDim2.new(0, mouse.X, 0, mouse.Y) + info.offset
38.             label.Position = newPosition
39.             sublimacao.BackgroundColor3 = Color3.new(0.827451, 0.827451, 0.827451)
40.             fusao.BackgroundColor3 = Color3.new(0.827451, 0.827451, 0.827451)
41.             solidificacao.BackgroundColor3 = Color3.new(0.827451, 0.827451,
                0.827451)
42.             liquefacao.BackgroundColor3 = Color3.new(0.827451, 0.827451, 0.827451)
43.             vaporizacao.BackgroundColor3 = Color3.new(0.827451, 0.827451, 0.827451)
44.         end
45.     end
46. end
47. -- Conectar eventos para cada TextLabel

48. for _, label in ipairs(screenGui:GetDescendants()) do
49.     if label:IsA("TextLabel") then
50.         label.InputBegan:Connect(function(input)
51.             if input.UserInputType == Enum.UserInputType.MouseButton1 then
52.                 onMouseButton1Down(label)
53.

```

```

54.     end
55. end)
56.
57. label.InputEnded:Connect(function(input)
58.     if input.UserInputType == Enum.UserInputType.MouseButton1 then
59.         onMouseButton1Up(label)
60.         sublimacao.BackgroundColor3 = Color3.new(1, 1, 1)
61.         fusao.BackgroundColor3 = Color3.new(1, 1, 1)
62.         solidificacao.BackgroundColor3 = Color3.new(1, 1, 1)
63.         liquefacao.BackgroundColor3 = Color3.new(1, 1, 1)
64.         vaporizacao.BackgroundColor3 = Color3.new(1, 1, 1)
65.     end
66. end)
67. end
68. end
69.
70. mouse.Move:Connect(onMouseMove)
71.

```

Fonte: O Autor (2023)

O *script* utilizado nos **Frame's**, serve para criar uma espécie de imã, em que ao se aproximar da área do frame, a Label cola na posição definida, esta é definida manualmente, sendo assim, cabe a você defini-la da forma que melhor lhe atender.

Atente-se para o comando “`label.Position = UDim2.new(0.63, 0,0.73, 0)`”, este é o responsável por definir uma coordenada para a **Label**, sendo assim, você deve encontrar a melhor posição de seu objeto de acordo com a propriedade **Position**, da **Label** a qual você selecionar. Sendo assim, acrescente o conteúdo do Script 5 para cada um dos **Frame's**, lembre-se de adequar a posição de acordo com cada um deles.

Script 5 – Verificação de Label Correta

```

1. local meuFrame = script.Parent
2. local TemLabel = meuFrame.TemLabel
3. local labels = {} -- Crie uma tabela para armazenar todas as labels que você
   deseja controlar
4. -- Adicione todas as labels que você deseja controlar à tabela "labels"
5. table.insert(labels, meuFrame.Parent:FindFirstChild("LabelFusao"))
6. table.insert(labels, meuFrame.Parent:FindFirstChild("LabelSolidificacao"))
7. table.insert(labels, meuFrame.Parent:FindFirstChild("LabelLiquefacao"))
8. table.insert(labels, meuFrame.Parent:FindFirstChild("LabelSublimacao"))
9. table.insert(labels, meuFrame.Parent:FindFirstChild("LabelVaporizacao"))
10. -- Função para ajustar a posição de uma label específica
11. local function ajustarPosicao(label)

```

```

12.  label.Position = UDim2.new(0.63, 0,0.73, 0)
13. end
14. -- Função para verificar se a label está dentro do frame
15. local function verificarLabelNoFrame(label)
16.  local labelScreenPosition = label.AbsolutePosition
17.  local labelSize = label.AbsoluteSize
18.  local frameScreenPosition = meuFrame.AbsolutePosition
19.  local frameSize = meuFrame.AbsoluteSize
20.  if (labelScreenPosition.X >= frameScreenPosition.X and
21.    labelScreenPosition.X + labelSize.X <= frameScreenPosition.X + frameSize.X and
22.    labelScreenPosition.Y >= frameScreenPosition.Y and
23.    labelScreenPosition.Y + labelSize.Y <= frameScreenPosition.Y + frameSize.Y)
24.  then
25.    -- A label está dentro do frame
26.    if(TemLabel.Value == false)then
27.      ajustarPosicao(label)
28.      TemLabel.Value = true
29.    else if (TemLabel.Value== true) then
30.      if(label.Position ~= UDim2.new(0.63, 0,0.73, 0))then
31.        if(TemLabel.Value==true and label.Name == 'LabelFusao')then
32.          label.Position = UDim2.new(0.743, 0,0.2, 0)
33.        else if(TemLabel.Value==true and label.Name == 'LabelSolidificacao')then
34.          label.Position = UDim2.new(0.166, 0,0.274, 0)
35.        else if(TemLabel.Value==true and label.Name == 'LabelLiquefacao')then
36.          label.Position = UDim2.new(0.065, 0,0.199, 0)
37.        else if(TemLabel.Value==true and label.Name == 'LabelSublimacao')then
38.          label.Position = UDim2.new(0.645, 0, 0.273, 0)
39.        else if(TemLabel.Value==true and label.Name ==
40.          'LabelVaporizacao')then
41.          label.Position = UDim2.new(0.407, 0,0.274, 0)
42.        end
43.      end
44.    end
45.    TemLabel.Value=false
46.  end
47. end
48. end
49. end
50. end
51. -- Conecte a função verificarLabelNoFrame ao evento MouseMove
52. game:GetService("UserInputService").InputChanged:Connect(function(input,
53.  gameProcessedEvent)
54.  if input.UserInputType == Enum.UserInputType.MouseMovement then
55.    for _, label in pairs(labels) do
56.      verificarLabelNoFrame(label)

```

```

56.     end
57. end
58. end)
59.

```

Fonte: O Autor (2023)

Por fim, tem-se os *script's* dos botões de verificação e de recarregamento da experiência, os quais fazem uso de posições fixas das **Label's**, dessa forma utilize a mesma dos *script's* anteriores. Para o botão de verificação utilize o código do Script 6.

Script 6 – Verificação de Label Correta

```

1. -- Conectar o evento MouseButton1Click do botão a uma função
2. local button = script.Parent
3. local solidificacao =
    script.Parent.Parent.ImageLabel:FindFirstChild("LabelSolidificacao")
4. local fusao = script.Parent.Parent.ImageLabel:FindFirstChild("LabelFusao")
5. local liquefacao =
    script.Parent.Parent.ImageLabel:FindFirstChild("LabelLiquefacao")
6. local vaporizacao =
    script.Parent.Parent.ImageLabel:FindFirstChild("LabelVaporizacao")
7. local sublimacao =
    script.Parent.Parent.ImageLabel:FindFirstChild("LabelSublimacao")
8. local pontos = 0
9. local pontuacao = script.Parent.Parent.Pontos
10. local function OnButtonClicked()
11.     pontos = 0
12.     if(solidificacao.Position == UDim2.new(0.232, 0,0.451, 0))then
13.         solidificacao.BackgroundColor3 = Color3.new(0, 1, 0)
14.         pontos = pontos+1
15.     else
16.         solidificacao.BackgroundColor3 = Color3.new(1, 0, 0)
17.     end
18.     if(fusao.Position == UDim2.new(0.311, 0,0.730, 0))then
19.         fusao.BackgroundColor3 = Color3.new(0, 1, 0)
20.         pontos = pontos+1
21.     else
22.         fusao.BackgroundColor3 = Color3.new(1, 0, 0)
23.     end
24.     if(liquefacao.Position == UDim2.new(0.59, 0,0.46, 0))then
25.         liquefacao.BackgroundColor3 = Color3.new(0, 1, 0)
26.         pontos = pontos+1
27.     else
28.         liquefacao.BackgroundColor3 = Color3.new(1, 0, 0)
29.     end
30.     if(vaporizacao.Position == UDim2.new(0.63, 0,0.73, 0))then

```



```

31.     vaporizacao.BackgroundColor3 = Color3.new(0, 1, 0)
32.     pontos = pontos+1
33. else
34.     vaporizacao.BackgroundColor3 = Color3.new(1, 0, 0)
35. end
36. if(sublimacao.Position == UDim2.new(0.431, 0, 0.857, 0))then
37.     sublimacao.BackgroundColor3 = Color3.new(0, 1, 0)
38.     pontos = pontos+1
39. else
40.     sublimacao.BackgroundColor3 = Color3.new(1, 0, 0)
41. end
42. pontuacao.Text = pontos
43. end
44. button.MouseButton1Click:Connect(OnButtonClicked)
45.

```

Fonte: O Autor (2023)

Já para o botão de recarregamento da experiência utilize o código do Script 7:

Script 7 – Verificação de Label Correta

```

1. -- Conectar o evento MouseButton1Click do botão a uma função
2. local button = script.Parent
3. local solidificacao =
    script.Parent.Parent.ImageLabel:FindFirstChild("LabelSolidificacao")
4. local fusao = script.Parent.Parent.ImageLabel:FindFirstChild("LabelFusao")
5. local liquefacao =
    script.Parent.Parent.ImageLabel:FindFirstChild("LabelLiquefacao")
6. local vaporizacao =
    script.Parent.Parent.ImageLabel:FindFirstChild("LabelVaporizacao")
7. local sublimacao =
    script.Parent.Parent.ImageLabel:FindFirstChild("LabelSublimacao")
8. local pontuacao = script.Parent.Parent.Pontos
9.
10. local function OnButtonClicked()
11.     solidificacao.Position = UDim2.new(0.166, 0,0.274, 0)
12.     solidificacao.BackgroundColor3 = Color3.new(1, 1, 1)
13.
14.     fusao.Position = UDim2.new(0.743, 0,0.2, 0)
15.     fusao.BackgroundColor3 = Color3.new(1, 1, 1)
16.
17.     liquefacao.Position = UDim2.new(0.065, 0,0.199, 0)
18.     liquefacao.BackgroundColor3 = Color3.new(1, 1, 1)
19.
20.

```

```
21. vaporizacao.Position = UDim2.new(0.407, 0,0.274, 0)
22. vaporizacao.BackgroundColor3 = Color3.new(1, 1, 1)
23.
24. sublimacao.Position = UDim2.new(0.645, 0, 0.273, 0)
25. sublimacao.BackgroundColor3 = Color3.new(1, 1, 1)
26.
27. pontuacao.Text = 0
28. end
29.
30. button.MouseButton1Click:Connect(OnButtonClicked)
31.
```

Fonte: O Autor (2023)

Ao construir essas duas experiências, você possuirá o conhecimento necessário para construir tanto no contexto 2D quanto no 3D. Assim, poderá utilizar os OVA construídos no contexto que melhor lhe atender, bastando utilizar sua imaginação e explorar novos conceitos.