Desenvolvimento com Motores de Jogos I

Aula 12 - Introdução à Interface com o Usuário no Unity

Apresentação

Olá, pessoal! Animados para mais uma aula de nossa disciplina de Desenvolvimento com Motores de Jogos I? Há muitos outros assuntos legais para aprendermos hoje! Como dissemos ao final da aula passada, chegou o momento de começarmos a utilizar interface gráfica em nossas cenas!

Nos jogos, precisamos sempre de uma interface para indicar ao usuário o que está acontecendo. Isso é parte da definição de jogos e não é diferente para jogos digitais. Em nosso caso, usualmente temos alguns componentes indicando informações importantes, como quantas chances restam ao jogador ou mesmo em qual fase ele se encontra.

Na aula de hoje, conheceremos o elemento Canvas, adicionado ao Unity em sua versão 4 e presente nele até hoje. Através desse elemento, podemos definir componentes de UI, do inglês User Interface, ou, em português, interface com o usuário. Conheceremos esse elemento e, também, os diversos componentes que o formam. Além disso, começaremos a conhecer os elementos da UI de fato, os quais incluem, por exemplo, textos e imagens.

Conheceremos, ainda, o EventSystem e o Rect Transform, que são partes importantes do desenvolvimento de interfaces com o usuário no Unity. Aprenderemos bastante sobre todos esses elementos de interface e finalizaremos a nossa aula utilizando esses conhecimentos para criarmos uma cena simples de Game Over. Nas próximas aulas, adicionaremos novos componentes de interface às nossas cenas!

Objetivos

Ao final desta aula, você deverá ser capaz de:

- Utilizar o objeto Canvas quando necessário for;
- Adicionar filhos ao objeto Canvas, principalmente imagens e textos;
- Compreender o funcionamento do EventSystem e do Rect Transform.

1. Adicionando um Texto à Cena Game Over

Aqui, missão dada é missão cumprida! Finalizamos a aula passada prometendo que adicionaríamos algo à nossa tela de Game Over, portanto, a aula de hoje será iniciada a partir desse ponto. Quem não lembra bem dos conteúdos estudados na aula passada, pode aproveitar e dar uma revisada neles. Quem está em dia, mas precisa do projeto, pode obtê-lo aqui! Todos prontos? Vamos começar!

Primeiramente, iniciaremos o nosso projeto no Unity e, então, abriremos a cena citada – a GameOver.unity, a qual se encontra na nossa pasta de Scenes, dentro dos nossos Assets, na aba Project. Ao abrir essa cena, nos depararemos com uma cena 2D vazia. Se dermos o Play em nosso jogo, veremos apenas uma tela azul, representando o fundo gerado pela câmera automaticamente.

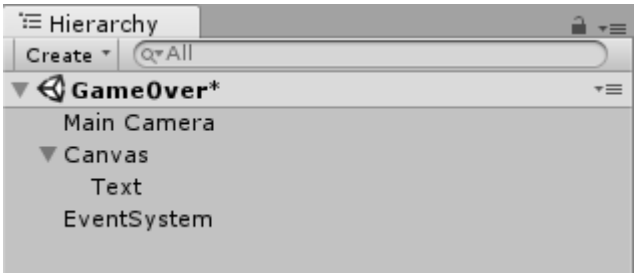
Para começar, poderíamos alterar essa cor de fundo. Quem lembra como podemos fazer isso? Aprendemos lá em nossa aula sobre câmeras! Essa aula, inclusive, será bem importante para o entendimento proveitoso da aula de hoje, uma vez que utilizaremos vários dos conceitos abordados lá. Que tal voltar um pouco e relembrar rapidamente os espaços de coordenadas nela vistos, além dessa troca de cor?

Relembrando a todos: para alterar a cor, vamos até o objeto Main Camera, no componente Camera, então, alteramos a propriedade Clear Flags para Solid Color e, em seguida, a propriedade Background para a cor desejada. Fiquem à vontade em suas escolhas! A minha foi a cor #5F6B8000, um tom de cinza. Acho que combina bem com as cores trabalhadas por nós! (Apesar de eu não entender nada sobre esquemas de cores lol)

Em seguida, precisamos adicionar o texto à nossa cena. Esse é o objetivo principal dela, pois, assim, podemos avisar ao jogador que ele perdeu, que suas chances acabaram, que é o fim de jogo! Game Over. Para adicionarmos esse texto, basta navegarmos ao menu GameObject e selecionar o caminho GameObject -> UI -> Text.

O menu UI contém diversos elementos de interface com o usuário disponibilizados aos desenvolvedores pelo Unity. Por meio desse menu, teremos acesso a todos os objetos que utilizaremos, nesta aula, ao tratarmos da interface. Começaremos pelo componente Text. A adição desse componente à nossa cena pode ser vista na **Figura 1**.

Figura 01 - Componente Text adicionado à cena. (E outros mais?)



Fonte: Captura de tela do Unity. Disponível em: https://unity3d.com/pt/. Acesso em: 14 de mar. de 2017.

Eita! Pelo visto, quando adicionamos o nosso texto à cena, diversas outras coisas apareceram também, não foi? Você pode observar isso na hierarquia, caso não tenha notado ainda. E não foi só um elemento. E esse elemento novo não tem transform. "Você não tinha dito que tudo tinha transform, professor? E agora?" "E o que são esses componentes?" Calma, pessoal! Agora, conheceremos melhor os elementos que compõem a interface com o usuário do Unity, começando pelo principal deles, o elemento Canvas! Assim, tudo ficará mais claro, quando voltarmos ao nosso exemplo.

2. Elementos de Interface com o Usuário no Unity

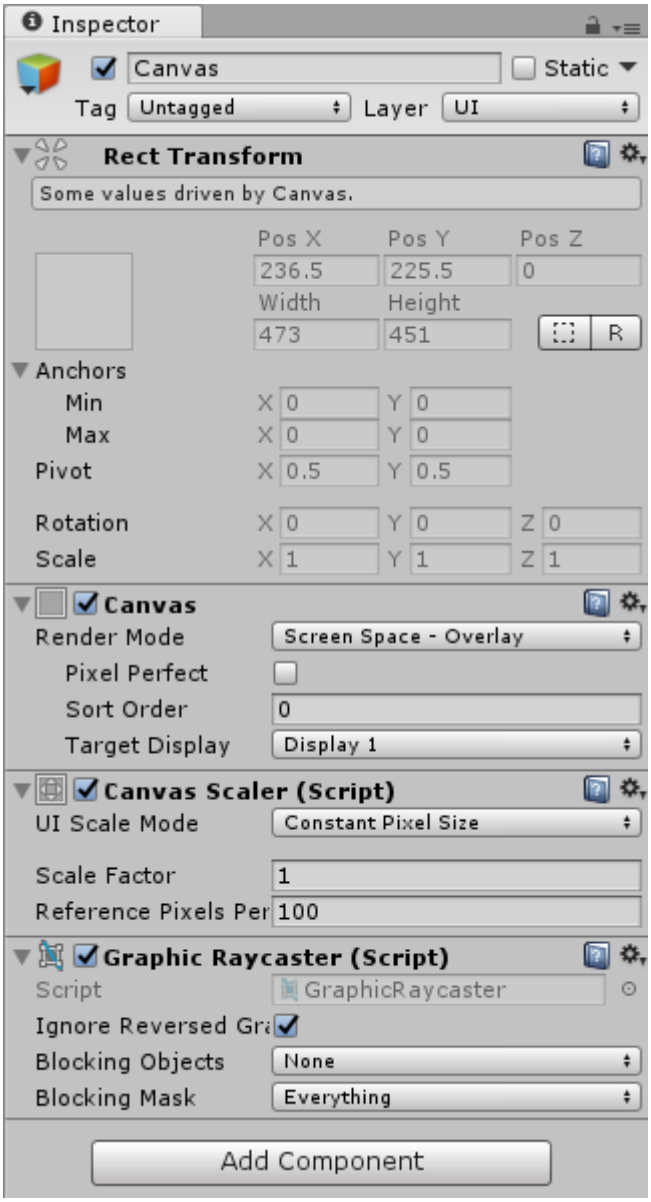
Sempre, ao adicionarmos um dos novos elementos de interface gráfica do Unity, criamos também, na cena, um Canvas e um EventSystem. Esses componentes são alternativas encontradas pelo Unity para facilitar o gerenciamento dos elementos de UI adicionados e de todos os eventos que são relacionados a tais elementos e podem surgir em seu ciclo de vida. Conheceremos melhor esses componentes auxiliares e entenderemos o motivo de essa solução ter sido escolhida.

2.1 O Elemento Canvas

O elemento Canvas é um GameObject criado automaticamente pelo Unity para guardar elementos de UI. Ele também pode ser criado manualmente pelo usuário, uma vez que uma cena pode possuir mais de um Canvas. Sendo assim, fica a critério do usuário escolher a utilização e posição de novos componentes desse tipo.

O elemento Canvas contém, desde sua criação, quatro componentes especiais: o *Rect Transform*, que substitui o transform; o componente *Canvas*, que contém algumas propriedades relacionadas ao próprio Canvas; o *Canvas Scaler* (script), que é um script gerenciador da escala do Canvas em relação à tela; e o *Graphic Raycaster* (script), que é um componente no qual há um script gerenciador de todos os raycasts necessários para a utilização de uma interface gráfica. Veja esses componentes na **Figura 2**.

Figura 02 - Componentes do GameObject Canvas.



Fonte: Captura de tela do Unity. Disponível em: https://unity3d.com/pt/. Acesso em: 14 de mar. de 2017.

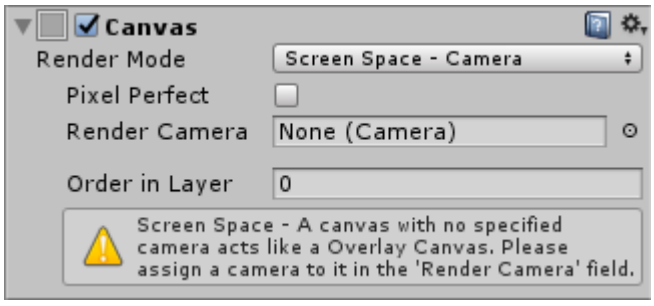
O primeiro componente visto ao selecionarmos o objeto Canvas é o componente Rect Transform. A fim de adaptar as possibilidades para algo mais específico do próprio Canvas e dos elementos de UI em geral, esse componente substitui o Transform clássico que temos em todos os objetos do Unity. Ainda possuímos escala e rotação, como vocês podem ver nas duas últimas linhas do componente, mas a propriedade referente à posição foi bem alterada. Detalharemos o Rect Transform a seguir, em sua própria seção.

O segundo componente visto é o componente Canvas. Ele apresenta as propriedades do próprio Canvas e sua configuração é primordial para que a nossa interface com o usuário seja exibida adequadamente, posicionada da maneira desejada.

A primeira propriedade do elemento, a **Render Mode**, indica qual será o espaço de coordenadas que utilizaremos para renderizar os objetos do Canvas. O primeiro deles, mais utilizado dentre todos, é o **Screen Space – Overlay**. Quando estamos utilizando esse modo para definir a nossa UI, ela é posicionada à frente de todos os elementos da cena, diretamente no espaço da tela de coordenadas, como uma camada de overlay. Ou seja, desenhamos todo o jogo e, em seguida, os elementos da UI.

Como geralmente as interfaces com os usuários se posicionam à frente de todo o jogo, o mais comum para elas é utilizar o valor Screen Space – Overlay para essa propriedade, a fim de poder exibir informações ao usuário sem que qualquer parte dela seja bloqueada por elementos do jogo. Esse modo traz mais três propriedades a serem configuradas. A primeira delas, Pixel Perfect, indica se os elementos de UI contidos nessa camada devem tentar se adequar, em perfeição de pixels, à tela. Isso pode criar uma visualização mais bem definida de sua UI como um todo. Em seguida, temos a propriedade Sort Order, que indica, como para os sprites, a ordem de desenho daquele Canvas em relação aos outros. Por fim, a propriedade Target Display indica, em setups de múltiplos monitores, em qual das telas disponíveis o Canvas deve desenhar os seus elementos.

Figura 03 - Canvas configurado no modo Screen Space – Camera.



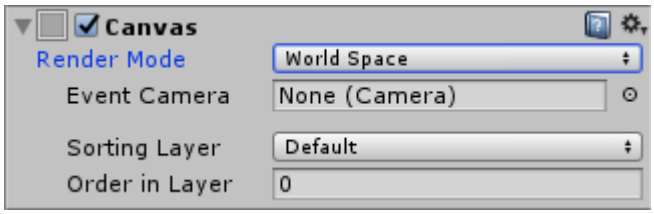
Fonte: Captura de tela do Unity. Disponível em: https://unity3d.com/pt/. Acesso em: 14 de mar. de 2017.

A segunda opção de **Render Mode** para o Canvas é a opção **Screen Space – Camera**, vista na **Figura 3**. Nesse modo, o Canvas será desenhado baseado em uma câmera e se posicionará sempre à frente para ela. Caso não escolhamos nenhuma

câmera, ele será considerado em modo **Screen Space – Overlay** e funcionará exatamente como vimos anteriormente. Caso uma câmera seja selecionada, ele se adequará e será desenhado na renderização específica dela.

Lembra de comentarmos, na aula sobre câmeras, que temos alguns casos viáveis no quais devemos utilizar mais de uma câmera? Para gerar um espelho, por exemplo? Pois é! Imagine que queremos colocar um menu a ser acessado a partir daquele espelho, como se o espelho fosse uma interface de toque. Podemos fazer isso utilizando o Canvas em relação àquela câmera do espelho. Basta configurá-la na propriedade Render Camera! Além disso, nesse modo, o Canvas tem as mesmas propriedades Pixel Perfect e Order in Layer.

Figura 04 - Canvas configurado no modo World Space.



Fonte: Captura de tela do Unity. Disponível em: https://unity3d.com/pt/. Acesso em: 14 de mar. de 2017.

O último **Render Mode** possível para o componente Canvas é o modo **World Space**, visto na **Figura 4**. Nesse modo, saímos do espaço de coordenadas da tela e passamos ao espaço de coordenadas do mundo. Ou seja, nosso Canvas agora não mais é renderizado por último, diretamente na tela ou por cima da câmera. Agora, ele passa a ser renderizado no mundo, como um objeto qualquer, podendo ser sobreposto ou mesmo sofrer alterações durante a cena. Você consegue imaginar algum lugar em que podemos utilizar esse tipo de Canvas? Direto na cena?

Não? Vejamos uma ideia, na **Figura 5**.

Figura 05 - Personagem do jogo Tibia falando “Stand still ya tasty snack”. O texto é renderizado no espaço do mundo onde o personagem está.

Fonte: Tibia by Cipsoft. Disponível em: http://www.tibia.com.. Acesso em: 14 de mar. de 2017.

Pronto! Podemos utilizar o Canvas no espaço do mundo, a fim de diálogos serem criados para personagens, ou mesmo elementos de interface, como o quadrado posicionado ao redor do inimigo, visto na **Figura 5**, serem colocados diretamente no mundo e acompanharem o personagem (ou não, depende de sua escolha).

Perceba que esse Render Mode tem também três propriedades. A primeira, *Event Camera*, indica qual será a câmera utilizada como geradora de eventos para aquele Canvas. Somente toques, cliques, etc. realizados a partir da visão da câmera selecionada, serão possíveis de serem detectados. É comum utilizar a própria *Main Camera* nesse componente, mas é possível trabalhar outras também, de acordo com a necessidade do seu jogo. As outras duas propriedades já foram estudadas por nós – *Sorting Layer* e *Order in Layer* indicam a ordem de renderização dos elementos desse Canvas em relação *aos outros elementos do mundo*, uma vez que agora eles se misturam. Você percebe a fala do personagem, na **Figura 5**, sobrepondo até mesmo a barra azul de mana do personagem? Pois é! Isso pode ser feito apenas alterando a ordem de desenho daquele Canvas específico.

Além do componente Canvas, que contém os três modos citados anteriormente e suas especificidades, temos também o script Canvas Scaler como um dos objetos padrão do Canvas. Esse script é responsável por controlar as alterações de tamanho no Canvas, visto que é comum o jogo sofrer alterações de tamanho de tela, se executado em computadores ou dispositivos móveis. A ideia de saber exatamente as especificações de destino funciona apenas para consoles!

Não precisamos detalhar muito esse componente, pois sua configuração inicial já é, usualmente, boa o bastante. Perceba, no entanto, que ele só funciona para quando o espaço do Canvas é a tela. Quando o Canvas passa a ser parte do mundo, ele é tratado como tal e não necessita de uma nova escala à medida que suas propriedades sejam alteradas.

Caso o Canvas esteja configurado para a tela, teremos três opções de escala para a UI. A primeira, **Constant Pixel Size**, mantém o tamanho, em pixels, constante. Com isso, à medida que a tela varia, o menu continua ocupando a mesma quantidade de pixels. Isso o fará variar de tamanho. Na segunda, **Scale with Screen Size**, a UI será definida a partir de uma resolução de referência e, então, se adaptará a novas resoluções de acordo com a base. Por fim, na **Constant Physical Size**, a UI terá um tamanho físico constante, variando de acordo com o dpi (dots per inch, ou pontos por polegada) da tela a fim de manter-se nesse tamanho, mesmo com variação de tela e resolução. Entenda bem essas três propriedades, para saber escolher qual delas utilizar ao desenvolver os seus menus.

Há alguns anos, era necessário, quando desejávamos adicionar um componente de interface ao nosso jogo, criar tudo manualmente, mesmo que utilizássemos o Unity . Os objetos em si deveriam ser posicionados de acordo com a câmera, para poderem aparecer em sua cena. Já os eventos, como cliques e toques, precisavam ser implementados individualmente, utilizando técnicas de raycasting para saber exatamente onde o toque aconteceu e se algum dos elementos de UI foi ativado.

Lembra de comentarmos, na aula em que apresentamos a câmera, sobre os sistemas de coordenadas e a importância das mudanças entre eles para podermos implementar a UI? Pois é! Era exatamente assim que acontecia. Recebíamos um clique no espaço da tela, convertíamos esse clique para o espaço do mundo, jogávamos um raio com a técnica de raycasting, a qual é o equivalente 3D da técnica de Line Casting estudada em aulas anteriores, para só depois, a partir das colisões detectadas pelo raycasting, saber qual elemento foi clicado e, então, ativar,

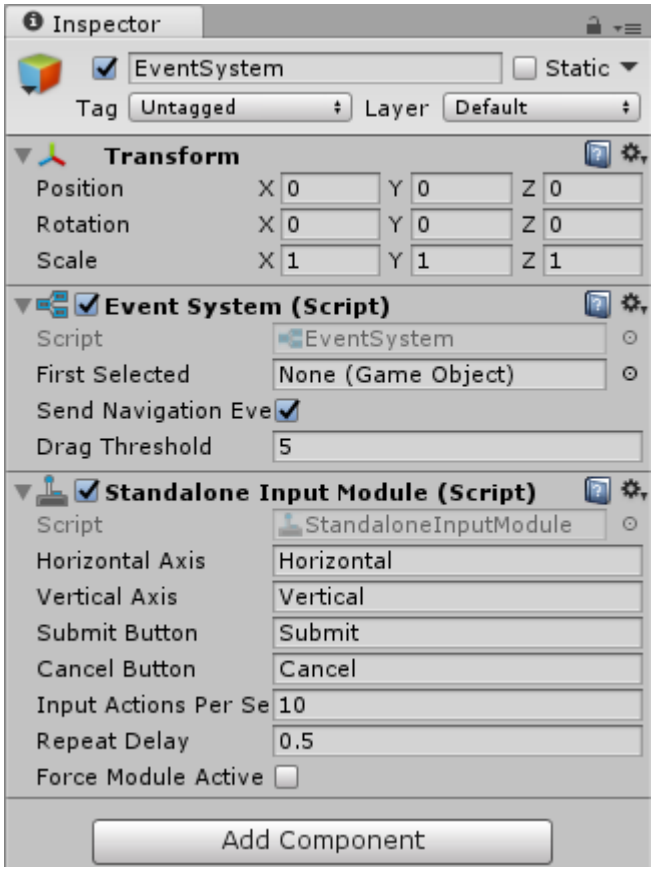
manualmente também, um script dele. Para evitar tudo isso, cada Canvas agora tem o seu próprio script de detecção de raycasting. Esse é o componente Graphic Raycaster.

E a utilização do resultado desse raycasting automático? Como fazemos? Bem, isso passa a ser responsabilidade do EventSystem, componente, estudado a seguir, que também é criado automaticamente quando adicionamos um elemento de UI à cena.

2.2 O Elemento EventSystem

Ao criarmos um Canvas em nossa cena, também é criado automaticamente um elemento chamado EventSystem. Esse elemento é responsável por lidar com eventos de UI de uma maneira mais simples, baseado nos resultados do componente de raycasting explicado anteriormente. Esse EventSystem traz também algumas configurações importantes a partir de seus componentes. A **Figura 6** exibe esses componentes para que possamos discuti-los em seguida.

Figura 06 - Componentes do elemento EventSystem.



Fonte: Captura de tela do Unity. Disponível em: https://unity3d.com/pt/. Acesso em: 14 de mar. de 2017.

O EventSystem, diferentemente do Canvas e dos objetos de UI, possui um Transform normal, contendo as mesmas propriedades dos regulares. Isso acontece devido ao fato de a posição desse objeto não importar. O que precisamos dele são os outros dois componentes, vistos na **Figura 6** – o Event System (script) e o Standalone Input Module (script).

O componente Event System é responsável por gerenciar todo o sistema de eventos do Unity. Ele é um script padrão do Unity, mas não detalharemos isso. O importante, no entanto, é saber que é necessário ele estar presente para gerenciarmos os eventos adequadamente. Outro detalhe interessante é a sua propriedade First Selected. Essa propriedade pode receber um elemento de UI qualquer, o qual, assim que a cena for carregada, será o ativo. Isso permitirá a você navegar entre os outros elementos utilizando os eixos e selecionar o elemento apenas apertando um botão de confirmação. Você consegue imaginar um lugar onde isso é importante? Vejamos a **Figura 7**.

Figura 07 - Menu principal do jogo Overwatch. A opção Play começa selecionada para ser possível navegar pelo menu utilizando o controle.

Fonte: Overwatch by Blizzard. Disponível em: http://gameuis.com/games/overwatch/. Acesso em: 14 de mar. de 2017.

Percebe a utilidade de possuirmos um componente inicial, já selecionado? Podemos, assim, navegar bem pelo menu, mesmo com um controlador, além de acessar rapidamente, apenas com o clique de um botão, a opção que já começa

selecionada por padrão. Simples, não?

Há também duas outras propriedades nesse componente que podem ser utilizadas ao longo de nossa navegação. A primeira delas, a propriedade Send Navigation Event, diz respeito ao envio de notificações (eventos) quando o menu for navegado. É possível, desse modo, alterar componentes a partir apenas desse trigger. A propriedade Drag Threshold, por sua vez, define uma área, em pixels, a partir da qual o arrasto de componentes se torna um evento válido. Isso é muito importante para evitar que um pequeno deslize do mouse, enquanto se clica em um componente, por exemplo, torne-se um evento de arrastar.

O último componente do EventSystem é o componente Standalone Input Module, o qual é responsável por configurar os controles a serem recebidos pelo sistema de eventos. Perceba haver, no script, uma definição de eixos e botões que serão aceitos para navegar pela UI. Esses eixos são exatamente iguais aos eixos definidos em nossa aula sobre Input.

Além dos eixos, há uma configuração da quantidade de ações que os menus podem receber por segundo. Esse valor é definido através da propriedade Input Actions Per Second. Isso evita que caso o usuário pressione uma tecla por um curto período, a navegação se torne difícil,. A configuração inicial é com o valor 10, indicando haver um delay de 1/10 de segundo entre uma ação e outra. Isso é, usualmente, bom o bastante.

Já a propriedade Repeat Delay indica qual é o atraso que há na repetição de uma ação, caso a tecla para realizá-la seja mantida pressionada ao longo de algum tempo. Essa delay é em segundos. O valor inicial de 0.5 é maior que o valor do Input Actions Per Second, fazendo, assim, a navegação no menu não ser afetada por esse delay.

Com isso, concluímos o estudo do módulo EventSystem. É bem importante entender as propriedades envolvidas nele, uma vez que isso facilitará a implementação de alguns detalhes da UI, quando você fizer uma para o seu jogo. Além disso, é importante entender como o EventSystem funciona, recebendo inputs e passando-os aos elementos de UI devidos.

2.3 O Componente Rect Transform

O último componente da UI que estudaremos antes de retornar à nossa tela de Game Over é o componente Rect Transform, presente em nosso Canvas e, também, em todos os elementos de UI a serem adicionados à nossa cena. Esse componente substitui o Transform e possui a mesma função dele – posicionar os objetos no espaço, com escala e rotação. Vejamos na **Figura 8**.

Figura 08 - Componente Rect Transform, utilizado em elementos da UI.

Fonte: Captura de tela do Unity. Disponível em: https://unity3d.com/pt/. Acesso em: 14 de mar. de 2017

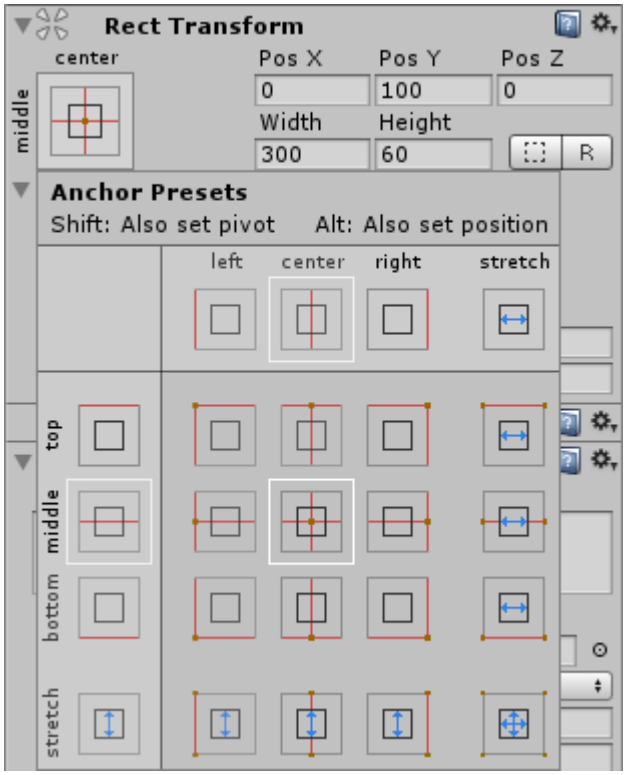
Para alguns elementos, o Rect Transform termina por ficar desabilitado. Um exemplo disso é o próprio Canvas quando configurado para ocupar o espaço da tela. Nessa configuração, o Canvas ocupará todo o espaço disponível, centrado no centro e sem qualquer giro em relação à tela. Se essas propriedades todas já estão definidas, não há motivos para configurar o Rect Transform e, por isso, ele é desabilitado, com uma mensagem indicando que o valor é controlado pelo próprio Canvas.

Para outros objetos, no entanto, isso é diferente. Podemos alterar esse componente a fim de posicionar adequadamente o objeto dentro do Canvas, por exemplo. Se observarmos o componente Rect Transform, na **Figura 8**, veremos que as suas propriedades são também relacionadas à posição, escala e rotação. Elas são, no entanto, bem alteradas em relação ao Transform normal. Para a escala e a rotação, os três valores (X, Y e Z) são mantidos. Para a posição, entretanto, precisamos entrar em mais detalhes.

O Rect Transform divide a posição em Pos X, Pos Y e Pos Z, da mesma maneira que a posição do Transform regular. Isso, no entanto, não é tudo. Há, também, a necessidade de configurar, logo abaixo, valores para Width e Height. Esses valores definem o tamanho do objeto e a quantidade do espaço de tela eles que ocuparão.

O posicionamento, porém, parte também da âncora utilizada para cada um dos componentes. No canto superior esquerdo do componente, temos uma imagem quadrada indicando qual a âncora, ou o ponto de pivô, como já vimos anteriormente, daquele objeto. Podemos clicar nesse elemento para serem revelados diversos presets de âncoras disponíveis. Selecionar um deles altera, de uma maneira mais visual, o ponto de âncora automaticamente, facilitando o posicionamento adequado. Veja, na **Figura 9**, exemplos desses presets para pontos de âncora.

Figura 09 - Presets disponíveis para pontos de âncora em um Rect Transform.



Fonte: Captura de tela do Unity. Disponível em: https://unity3d.com/pt/. Acesso em: 14 de mar. de 2017

É interessante notar que, quando a âncora é redefinida, o Unity se encarrega de alterar todos os outros valores, para manter o objeto posicionado onde estava. Assim, mesmo alterando as propriedades do objeto, este se mantém onde o posicionamos visualmente. Perceba que você ainda pode alterar os valores manualmente, fazendo o objeto se deslocar. Note, também, ser possível utilizar a tecla Shift, a fim de alterar também o pivô do elemento, ao selecionar um dos presets. Você pode, além disso, utilizar o alt para alterar também a posição do elemento.

Outro detalhe importante é que essa ancoragem é realizada em relação ao objeto pai do objeto que está sendo ancorado. Se o objeto for diretamente filho do objeto Canvas, ele será ancorado ao Canvas, mas há também casos nos quais uma UI é composta por diversos objetos, e estes precisam ser ancorados entre eles. Nesses casos, é mais importante ainda lembrar que a ancoragem é feita em relação ao pai, e não diretamente ao Canvas.

Com isso, concluímos todo o conteúdo básico sobre as User Interfaces, ou as interfaces com o usuário no Unity. Ainda há muito mais o que aprendermos, além de diversos componentes que podemos utilizar, cada um com suas especificidades. No entanto, agora não as detalharemos. Voltaremos à nossa cena e construiremos o nosso Game Over baseado em dois elementos de interface. As características deles, porém, serão assuntos da próxima aula.

3. Finalizando a Cena de Game Over

Conhecemos, então, todos os elementos a serem adicionados em nossa cena de Game Over quando houver adição de um texto. Agora, podemos partir para a modificação e o posicionamento adequado dos elementos que de fato irão compor a nossa cena de Game Over.

3.1 Adicionando um Texto à Cena

A primeira parte é a alteração da cor e do tamanho do texto, além de seu conteúdo. Para fazer isso, basta clicar no componente Text, adicionado como filho do Canvas, e alterar essas propriedades no componente Text (script) contido nele. A propriedade Text indica qual o texto que utilizaremos para esse elemento. A propriedade Color indica a cor. Alterem o Text para “Game Over” (sem as aspas) e a Color para branco, #FFFFFFFF. Com isso, já devemos ter um texto Game Over aparecendo na tela. Esse texto, no entanto, estará em uma posição não adequada e será bem pequeno. Precisamos alterar isso também!

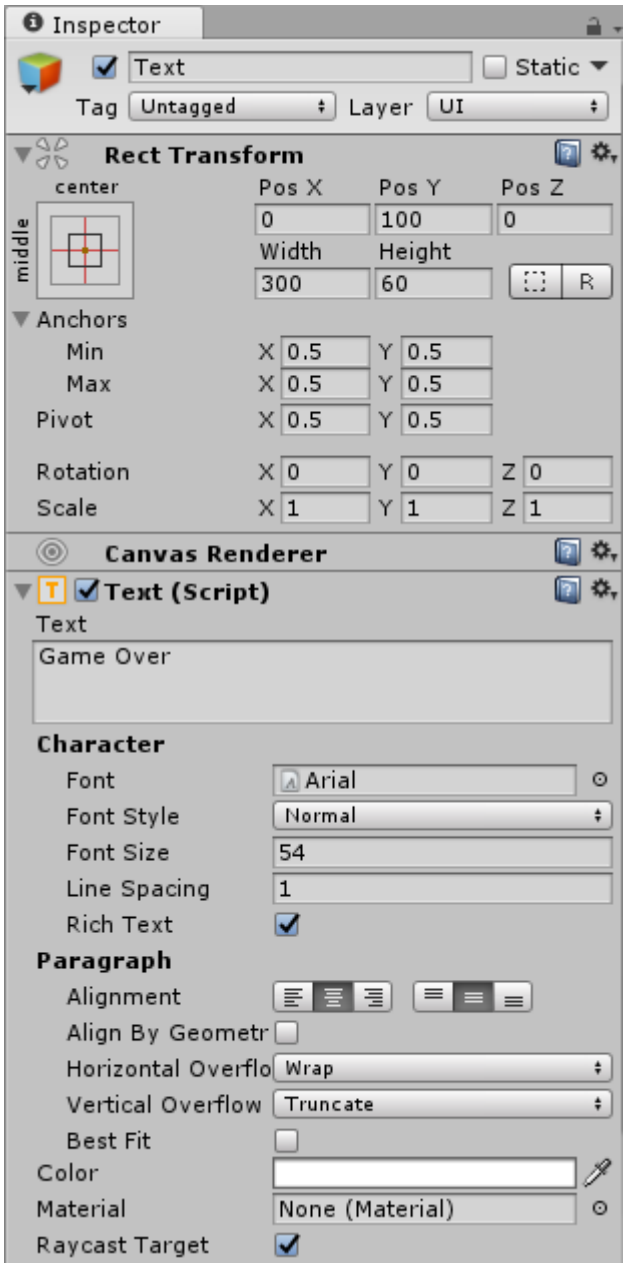
Para alterar o tamanho do texto, podemos alterar a propriedade Font Size, da mesma maneira como utilizamos em editores de texto regular. Diversas outras propriedades também serão similares às que já conhecemos lá da disciplina de Introdução à Tecnologia da Informação. Altere a Font Size para 54.

O que aconteceu? Sumiu? E agora?

Calma! Isso acontece, pois o texto escrito, com a fonte escolhida, não cabe dentro do espaço criado para o Text. Para aumentar o tamanho que pode ser ocupado por esse componente, basta alterarmos o Width e o Height, do componente Rect Transform. Utilizando o Width 300 com o Height 60 e a Pos Y 100, conseguimos adequar o texto a uma posição interessante da tela, também em um tamanho bom.

Perceba, ainda, que a âncora utilizada é a middle/center, ou seja, totalmente centralizada. Isso é importante, pois, como dissemos anteriormente, os valores de todos os componentes se alteram à medida que nós alteramos a âncora de um objeto. Esses valores passados funcionarão para uma âncora centralizada. Além disso, os valores de alinhamento também devem ser alterados para centralizados vertical e horizontalmente. A **Figura 10** mostra como ficarão as propriedades finais do texto de Game Over.

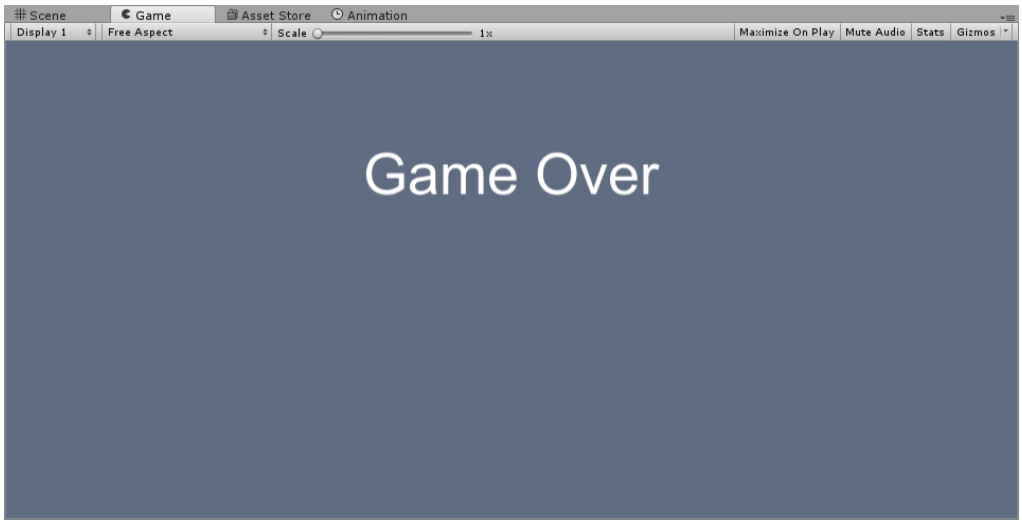
Figura 10 - Propriedades finais do texto de Game Over a ser utilizado na cena de Game Over.



Fonte: Captura de tela do Unity. Disponível em: https://unity3d.com/pt/. Acesso em: 14 de mar. de 2017

O texto, na tela, se você tiver utilizado as mesmas propriedades e a mesma cor de fundo que citamos anteriormente, ficará igual ao visto na **Figura 11**.

Figura 11 - Texto de Game Over posicionado na tela de fim de jogo.



Fonte: Captura de tela do Unity. Disponível em: https://unity3d.com/pt/. Acesso em: 14 de mar. de 2017

Com isso, já temos alguma informação ao usuário. Quando ele cair e perder as duas chances possuídas, logo saberá o que aconteceu: Game Over. Mas essa tela ainda está muito vazia. Vamos adicionar também uma imagem de fim de jogo para ela ficar mais interessante?

3.2 Adicionando uma Imagem à Cena

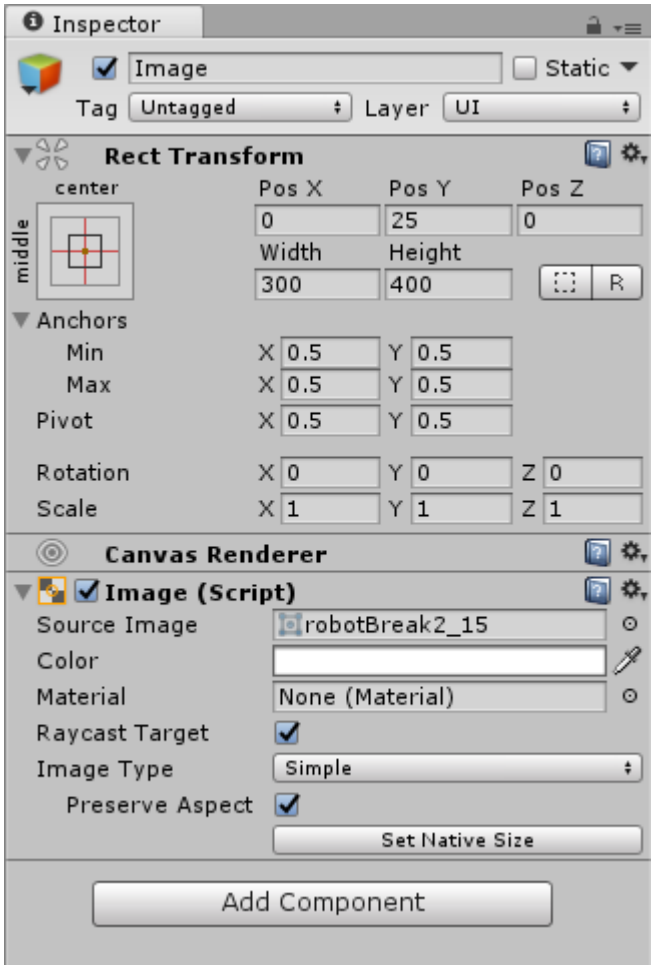
Para adicionarmos uma imagem à cena, podemos fazer do mesmo modo como fizemos anteriormente para o texto: GameObject -> UI -> Image. Perceba que o Unity criará automaticamente a nova imagem como filha do Canvas. Isso é interessante, pois não precisamos nos preocupar com a reconfiguração de qualquer Canvas, se realmente buscamos utilizar o mesmo Canvas. Caso quiséssemos utilizar um Canvas diferente, precisaríamos adicionar antes o novo Canvas à cena e, então, adicionar a imagem diretamente como um novo objeto filho daquele Canvas.

Agora que temos o GameObject Image em nossa cena, escolheremos o componente a ser utilizado. Acho interessante o último frame da animação de quebra representar o fim de jogo, uma vez que ele representa a perda de chances de nosso robô. Para escolher essa imagem, selecione o objeto Image e, na propriedade Source Image, escolha o robotBreak_15. Com isso, o nosso robozinho quebrado já aparecerá na tela, porém, bem pequeno.

Uma das opções que existem no Image (script), componente padrão das imagens em UI, é a opção Set Native Size. Ao escolhê-la, a imagem será transformada em seu tamanho original. Nesse caso, no entanto, esse tamanho é grande demais! Alteraremos, então, as propriedades da imagem, para que ele fique em uma posição e escala adequadas. Alteraremos o Width e o Height da imagem, respectivamente, para 300 e 400, valores correspondentes exatamente à metade do tamanho da imagem, uma vez que ela é 600 x 800, lembra?

Em seguida, já com a imagem no tamanho adequado, alteraremos a posição em Y para 25, indicando que ela ficará apenas um pouco acima do centro. Com isso, a nossa imagem já estará posicionada e teremos a nossa tela de Game Over concluída. A **Figura 12** mostra as propriedades do GameObject Image, e a Figura 13, a tela de Game Over após a finalização.

Figura 12 - Propriedades do GameObject Image adicionado à tela de Game



Fonte: Captura de tela do Unity. Disponível em: https://unity3d.com/pt/. Acesso em: 14 de mar. de 2017

Figura 13 - Tela de Game Over concluída, exibindo uma imagem de nosso personagem e um texto.

Fonte: Captura de tela do Unity. Disponível em: https://unity3d.com/pt/. Acesso em: 14 de mar. de 2017

Assim, concluímos a nossa aula de hoje, relativa à introdução à interface com o usuário no Unity. Ainda há muito o que discutirmos sobre esse assunto, portanto, aproveitaremos as próximas aulas para detalhá-lo mais, principalmente construindo uma interface para as cenas de nosso jogo em si! Legal, não? Pois nos encontramos lá! o/

Leitura Complementar

UI Canvas - https://unity3d.com/pt/learn/tutorials/topics/user-interface-ui/ui-canvas?playlist=17111

UI Rect Transform - https://unity3d.com/pt/learn/tutorials/modules/beginner/ui/rect-transform?playlist=17111

UI Events and Event System - https://unity3d.com/pt/learn/tutorials/topics/user-interface-ui/ui-events-and-event-triggers?playlist=17111

Resumo

Na aula de hoje, conhecemos a interface com o usuário no Unity e a maneira como podemos utilizar alguns de seus principais componentes para desenvolver tais interfaces. Conhecemos o elemento Canvas e toda a sua importância no posicionamento de elementos de UI. Estudamos todas as opções disponibilizadas para nós, por esses elementos, de escolhas de propriedades e vimos quando podemos utilizar cada uma delas, focando no conhecimento para desenvolvimento de jogos diversos.

Em seguida, estudamos os componentes Rect Transform e EventSystem. O primeiro é responsável por substituir o Transform clássico em elementos de interface com o usuário, facilitando, assim, o posicionamento e a adequação desses elementos. Já o segundo é responsável por gerir os eventos relacionados às interfaces do Unity e por lidar com Inputs adequadamente, configurável através dele mesmo para garantir uma boa fluidez às interfaces que respondem a eventos.

Por fim, voltamos à nossa cena de Game Over e adicionamos a ela um texto e uma imagem de nosso personagem, indicando ao usuário qual é a razão de ele ter chegado até aquela tela – fim de jogo! Não avançamos tanto em nosso projeto hoje, pois ficamos mais na teoria, mas aqui está o arquivo contendo tudo que fizemos até o momento.

Prosseguiremos com esse assunto na aula seguinte, começando a utilizar de fato os eventos dos quais falamos do EventSystem e construindo novos objetos capazes de interagir com o usuário.

Autoavaliação

1. O que é e qual a importância do Canvas para as UIs?
2. Quais os três modos possíveis para o Canvas? Pense em um exemplo de utilização para cada um deles.
3. Para que serve o EventSystem?
4. Como podemos configurar os Inputs que serão utilizados para interagir com a UI?
5. Quais as facilidades trazidas pelo Rect Transform em relação ao Transform?

Referências

Documentação oficial do Unity - Disponível em: https://docs.unity3d.com/Manual/index.html

Tutoriais oficiais do Unity - Disponível em: https://unity3d.com/pt/learn/tutorials

RABIN, Steve. **Introdução ao Desenvolvimento de Games**, Vol 2. CENGAGE.