Hero Dressup - Starter Kit

This is a starter kit to make a dressup/customization game.

Table of Contents

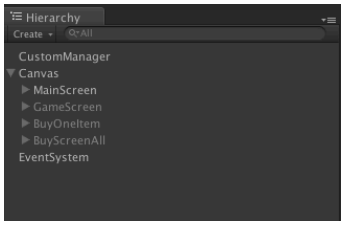
Getting Started

The first step you need to take is import the package in a new project.

- Once you have done that: locate the Main Scene (Assets/Scenes/Main.unity) and open it.

- Run it to see it working. The project is compatible for Unity 4.6 to 5+.

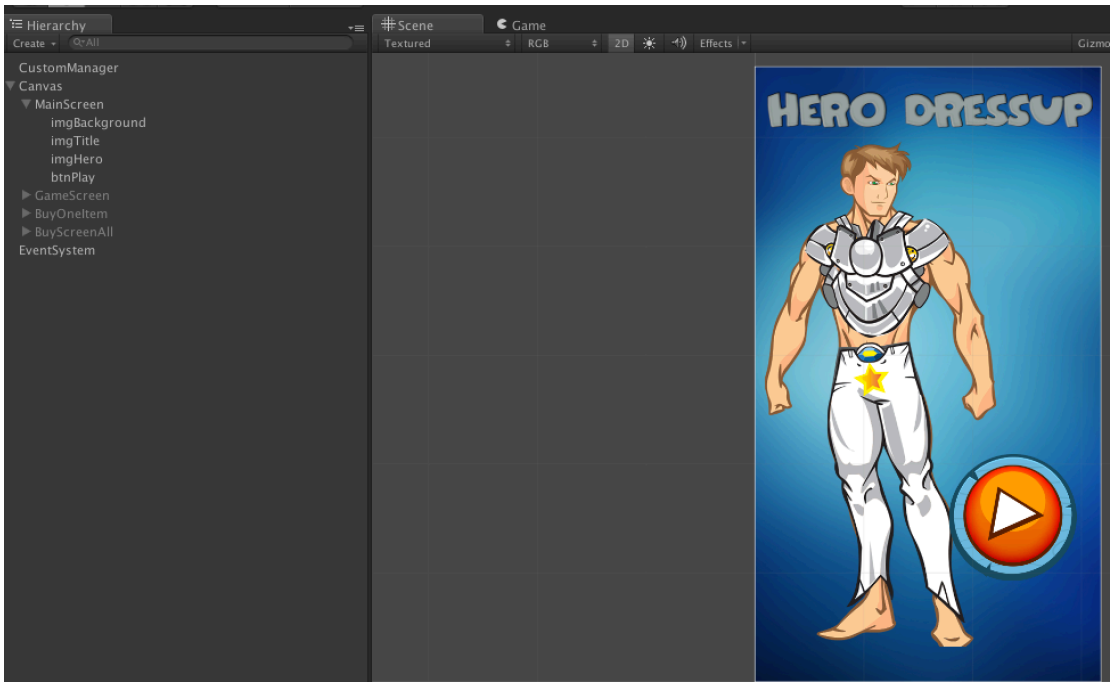
The Hierarchy



The scene is divided into different panels. There is a **CustomManager** Object which handles the clicks and all the customization functions.

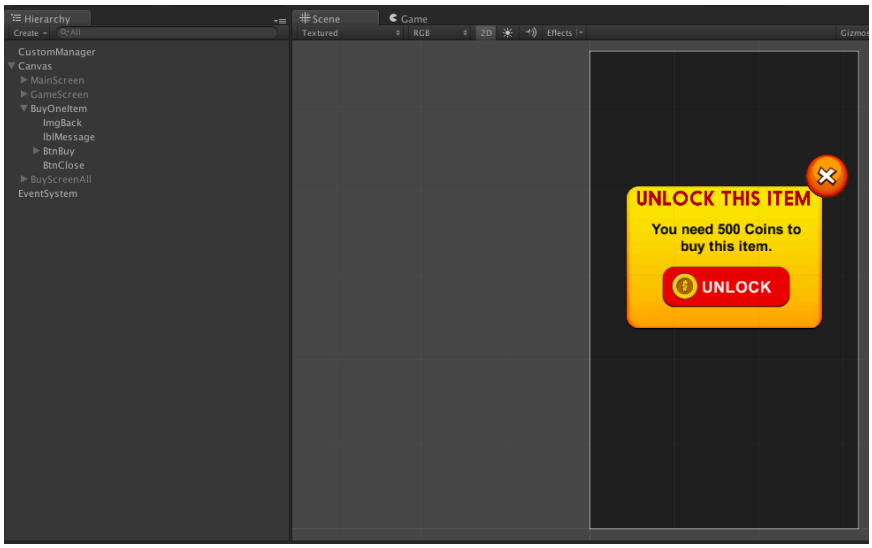
There is a 2D **Canvas** with four panels (screens) and an **Event System** object to handle 2D Clicks of Buttons.

The Main Screen Panel



The main screen panel is fairly basic with only a play button (btnPlay) that calls CustomManager.startGame function.

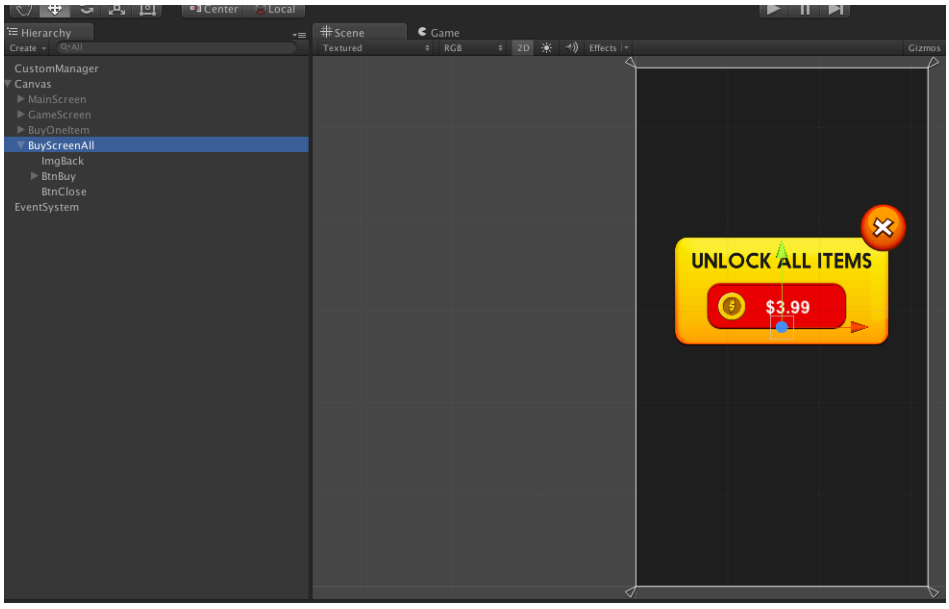
The Buy One Screen Panel



This panel appears on top of the Game Screen when a locked item is clicked on.

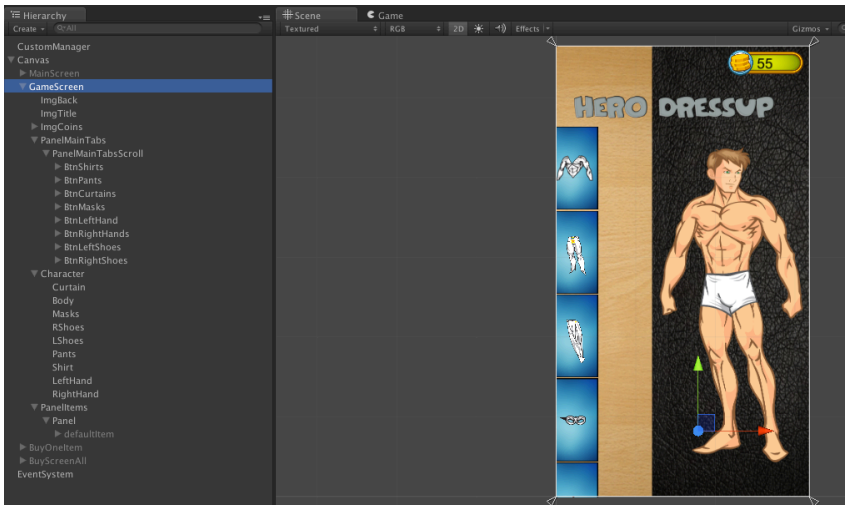
It contains two buttons (btnBuy, btnClose) that call the CustomManager.unlockThisItem and CustomManager.crossButtonPressed functions.

The Buy All Screen Panel



Similarly, the buy all screen panel has two buttons. The btnBuy calls CustomManager.buyAll function.

The Game Screen Panel



The Game Screen Panel is the main screen that contains the scroll view (with buttons), the coins label, the character and body parts.

Under the Character empty game object, you'll find different body parts gameobjects with image component attached. They are placed in the positions where all sprites of that category fall in place.

There are two panels to implement the scroll view (PanelMainTabs, PanelMainTabsScroll). The scroll rect component is on the PanelMainTabs.

All the buttons in the PanelMainTabsScroll (BtnShirts, BtnPants) call CustomManager.menuButtonClicked with a reference of the button passed.

The Code

There are two scripts that are most important. CustomItem.cs and CustomManager.cs

CustomItem is a class structure that is used in CustomManager.cs which is derived from MonoBehavior and assigned to the game object (Custom Manager).

CustomManager.cs

Functions:

```
    //Buy All function is called from the BuyAll Panel on the click of $3.99 button
    public void buyAll()

    //Cross button is called from the Buy All/ Buy One Panels to close them
    public void crossButtonPressed()

    //Main function called when the user presses the Play button from the MainScreen
panel
    public void startGame()

//Start function is called at the start of the game when the user is on MainScreen
    void Start()

    //Load data function is very important as it creates the different customitem objects
and adds them to the lists
    void loadData(string type, List<CustomItem> lst, bool withNull)

    //This function is called whenever the menu button is clicked from the GameScreen
    public void menuButtonClicked(Button thisButton)

    //The regenerate button is used as a replica of the above function
    public void regenerate(string btn)

    //This function is called from the Buy One screen when the user has enough coins t
o purchase the item
    public void unlockThisItem()

//This function is called from the Main Screen when an item is clicked
    public void itemClicked(Button thisButton)

    //function to add coins to the users Player prefs
    public void addCoins(int c)

    //function to update coins label everywhere
    public void updateCoins()
```

```
    //changes sprites of all buttons to inactive sprite
    void switchoffButtons()

    //A crucial function to instantiate items buttons when a menu is selected.
    //Buttons are generated according to the list of custom items passed
    void generateItems(List<CustomItem> lst)

    //All generated items objects are destroyed before new are created
    void destroyItems()
```

Data members

```
//Declaring Images of Main Menu Buttons
    public Sprite imgOnButton;
    public Sprite imgOffButton;

    //Panels of Buy One and Buy All Items
    public GameObject buyScreenOne;
    public GameObject buyScreenAll;

    //Coins Text - array because it can be on more than one panel
    public Text[] lblCoins;

    //Very important -
 currentScreen string is used to determine which screen is on top of the screen -
 don't enable the GameScreen in the start.
    //Always start the game with MainScreen enabled and all other disabled
    public string currentScreen;


    //Default Item - to be used for every item in the menu
    public GameObject defaultItem;

    //A string to store which menu is selected at the time
    string selectedMenu;

    //A simple text label in the buy one screen to inform the user how much coins are re
quired.
    public Text lblMessageBuyOne;

    //Current item selected
    int currentId = 0;

    //Selected Customitem
    public CustomItem currentItem;

    //Last item button clicked
```

```
    public Button lastItemClicked;

    //The actual game objects (with image script on them) on the scene -
the images will be set on these objects
    public Image objShirts;
    public Image objPants;
    public Image objRShoes;
    public Image objLShoes;
    public Image objCurtain;
    public Image objMasks;
    public Image objLeftHands;
    public Image objRightHands;

    //List of type CustomItem to store the items of different types
    List<CustomItem> Shirts = new List<CustomItem>();
    List<CustomItem> Pants = new List<CustomItem>();
    List<CustomItem> RightShoes = new List<CustomItem>();
    List<CustomItem> LeftShoes = new List<CustomItem>();
    List<CustomItem> Curtains = new List<CustomItem>();
    List<CustomItem> Masks = new List<CustomItem>();
    List<CustomItem> LeftHands = new List<CustomItem>();
    List<CustomItem> RightHands = new List<CustomItem>();

    //An array of menu buttons that are used -
inorder to make selected button active and others inactive
    public Button[] buttons;

    //Different Panels in the scene
    public GameObject mainPanel;
    public GameObject gamePanel;
```

CustomItem.cs

This is a data structure class used to store the details of the item.

Functions

```
//Prices of every type of object are set here.
    int getCategoryPrice(string categoryName)

    //This function returns how many objects from the current category are unlocked
    int categoryLocked(string categoryName)

    //The overloaded constructor
    public CustomItem(int id, string category, Sprite spr, int price, bool locked)

    //Returns the status of the item
    public bool isLocked()
```

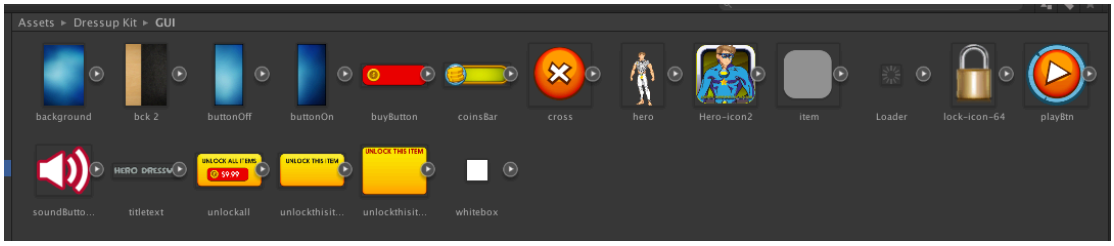
Data members

```
    //The identifier of the item
    public int id;

    //The category id
    public string categoryId;

    //The sprite associated with the item
    public Sprite spr;

    //The price of item
    public int price;
```

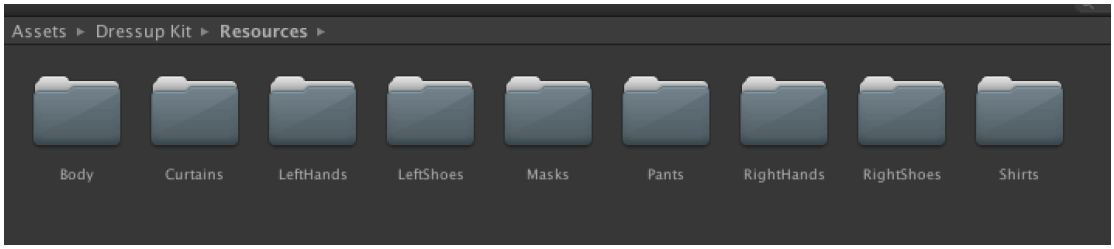
Reskin Guide/Files Structure

The files are stored in separate folders for your convenience.

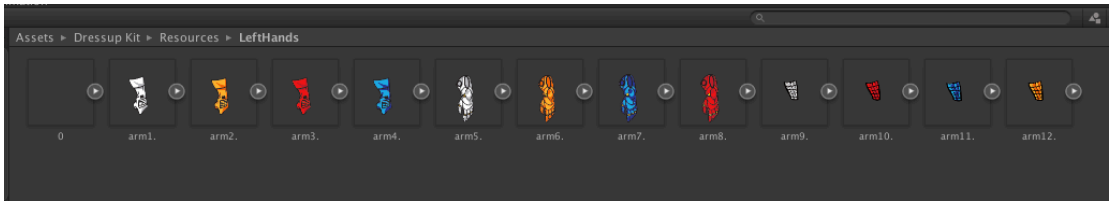
To edit the GUI, you need to change the sprites in the GUI folder.



To edit the images, you need to look in the Resources folder.



Every folder in the Resources folder has sprites for that particular category.



Support

Feel free to contact us at for Support at:

oneideaoneteam@gmail.com

Licensed royalty free music (Crusade) from: **incompetech.com**