Balloon Simulator

ENGINEER'S MANUAL

Acknowledgements

This project was designed with love and care by the following members of Team NoName

Vasyl Onufriyev - Project Manager & Developer

Jacob Potter – Developer

Cody Hawkins – Developer

Nelson Su – Developer

Logan Pettit – Developer

Table of Contents

Introduction

Thank you for downloading and using Balloon Simulator as part of your classroom's curriculum. We hope that you will find our software easy to use and reliable. This software was designed as part of a classroom project in CS4500 in the Fall of 2019, and released for public use by its developers.

What is Balloon Simulator?

Balloon Simulator is an open source project with the objective of providing a product to schools that is easily accessible and supports the advancement of S.T.E.M education in primary and secondary education. The goal of Balloon Simulator is to give students an insight into the physical properties of matter and physics by simulating a balloon filled with helium, whose properties can be tweaked and recorded to look for relationships between various properties of geometric shapes and physical forces.

How do I get started?

Getting started with balloon simulator is easy. Simply download one of our precompiled packages, extract it, and run it.

Balloon simulator comes preconfigured for various grade levels. These are as follow:

K – 2

- Color wheel that adjusts balloon color
- Radius slider that adjusts the balloon's radius when adjusted

3 – 5

- Everything from K – 2 EXCEPT color wheel PLUS
- Data boxes which display the metrics about the balloon

6 – 8

- Everything from 3 – 5 PLUS
- Graphing functions that allow plotting points on a graph
- Wind Slider which allows an additional force to be applied to the balloon

What if I want to customize what is available/visible?

Go to the location where you downloaded this package, go to the folder named *BalloonSim_Data*, then go to the folder named Config. Inside, there is a file named *config.json*. Inside, you can adjust which elements will be accessible to the users. Simply adjust the values inside to your desired values. Refer to configuration information for details about each value.

Full path:
%path_to_downloaded_package_folder/BalloonSim_Data/config/config.json

Configuration Information

In this section, there will be information provided on each value of the configuration file. The configuration file can be located at:

%path_to_downloaded_package_folder/BalloonSim_Data/config/config.json

Section 1: User Interface Elements Configuration

Section 1.1: Color Area Configuration

“colorWheel” Possible Values:

true - Enables the color area

false - Disables the color area

The configuration file contains values that reference the color wheel option to change the color of the balloon. The *colorWheel* value is set to true by default for the kindergarten – second simulation configuration. All other configurations have a default of false, turning the feature off.

Section 1.2: Radius Slider and Buttons Configuration

The configuration file contains values that reference the radius sliders properties named *minRadius*, *maxRadius*, and *radiusSlider*. Starting with the *radiusSlider*, the value can be true or false. A true value will provide the radius slider UI in the simulation and is set to true for every configuration. A false value will remove the radius slider from the UI and disable any change in radius of the balloon from the slider. The *minRadius* value is set to a minimum of 50 meters for a realistic visual when comparing the balloon to the rope. The *minRadius* value can be changed to anything greater than 50 and less than 400 with regard that the *minRadius* value is less than the *maxRadius* value. The *maxRadius* value has similar constrains, the *maxRadius* can be anywhere greater than 50 and less than 400 with regard that the *maxRadius* is greater than the *minRadius*. In the case that the *minRadius* value is greater than the *maxRadius* value or, the *maxRadius* value is less than the *minRadius* value, the default values will override the changes and set *minRadius* to 50 and *maxRadius* to 400. The UI also provides inflate and deflate buttons for incremental increase of the balloon radius. A false value will disable this feature.

Section 1.2.1 Radius Slider Configuration

“radiusSlider” Possible Values:

true - Enables the radius slider

false - Disables the radius slider

Details:

When *radiusSlider* is set to false it will disable the radius slider UI element but, allow for you to still change the balloon radius by using the inflate and deflate buttons if they are set to true. If the *InflateDeflateButton* and *radiusSlider* properties are set to false, the

switch box to switch between changing the radius and changing the wind will be disabled and only the wind slider will be available for user. This feature is by default set to true for all configurations of the simulation and values are in meters.

“minRadius” Possible Values:

Whole number values 50 – 399

Constraint:

Must be less than maxRadius

Details:

Changes the minimum radius of the balloon in meters.

“maxRadius” Possible Values:

Whole number values 51 – 400

Constraint:

Must be greater than minRadius

Details:

Changes the maximum radius of the balloon in meters.

Section 1.2.2 Inflate and Deflate Buttons Configuration

“inflateDeflateButton” Possible Values:

true - Enables the inflate / deflate buttons

false - Disables the inflate / deflate buttons

Details:

Disabling the *inflateDeflateButton* property will cause the inflate and deflate buttons to be removed from the UI. The radius slider will still be available for the user to change the radius of the balloon if it is set to True. Giving the *InflateDeflateButton* property a value of false and the *radiusSlider* property a value of false will cause the switch box UI element for changing between the wind and radius sliders to disappear and only the wind slider will be available for the user. This feature is by default set to true for all configurations and values are in meters.

“inflateIncrement” Possible Values:

Whole number

Constraint:

>= 1 AND < *maxRadius*

“inflateIncrement” Possible Values:

Whole number

Constraint:

>= 1 AND < *maxRadius*

Details:

The configuration file contains the values *inflateIncrement* and *deflateIncrement*, which refer to the change in meters of the radius. Each time either button is pressed it will change the value of the radius to the set values of *inflateIncrement* or *deflateIncrement*. The default value provided is 10 which corresponds to 10 meters. These values can be customized however the user likes with a few constraints. The value of *inflateIncrement* and *deflateIncrement* must be greater equal to 1 and less than the *maxRadius* value.

Section 1.3: Data Box Configuration

“dataBox” Possible Values:

true - Enables the data box

false - Disables the data box

Details:

The databox feature provides real time data to the user from changes in the physics of the balloon through use of the sliders or buttons. By default, the *dataBox* property is set to false for kindergarten – second grade configuration and set to true for the rest of the configurations.

Section 1.4: Wind Slider Configuration

“windSlider” Possible Values:

true - Enables the wind slider

false - Disables the wind slider

Details:

The wind slider applies wind to the balloon changing the data in the databox dynamically and providing a visual for the wind. Giving the *windSlider* a false value will cause the wind slider to be removed from the user’s UI and the switch box feature for switching between the radius slider and wind slider will be removed as well. Only the radius slider, inflate button, and deflate

button will be provided they all have a value of true. By default, the *windSlider* feature is set to true for only the sixth – eighth grade configuration.

“minWindSpeed” Possible Values:

Whole number values 0 – 4

Constraint:

Must be less than maxWindSpeed

Details:

Increases minimum wind speed of balloon in meters/second

“maxWindSpeed” Possible Values:

Whole number values 1 – 5

Constraint:

Must be greater than minWindSpeed

Details:

Increases maximum wind speed in meters/second

Section 1.5: Graph Configuration

“graph” Possible Values:

true - Enables the data collection/display area

false - Disables the data collection/display area

Details:

The graph provides real time recording for users to track the current physics data at the moment of pressing the record button in a graphical format. Providing a true value for *graph* will show the graph in the bottom right corner of the users UI. Disabling the graph will remove the graph from the users UI. By default, the graph is set to true for only the sixth – eighth grade configurations.

“recordButton” Possible Values:

true - Enables the data collection button

false - Disables the data collection button

Constraint:

graph must be true to be visible

Details:

The record button allows users to record real time physics data of the balloon on the graph feature. This features value of true is dependent on the graph property also being true. Disabling this feature will cause the graph to be disabled and both features removed from the UI. If this feature is enabled while the graph is disabled, the *recordButton* will be relabeled from record to export and the data from pressing export will be exported to a CSV (comma separated values) file. If the graph and *recordButton* are both set to true, the real time data will be provided in a visual graph that can be magnified and exported to a CSV. By default, the graph and record button features are set to true for only the sixth – eighth grade configurations.

"csvExportPath" Possible Values:

Export path string starting at the base of the program's installation

"imageExportPath" Possible Values:

Export path string starting at the base of the program's installation

Testing

Section 1: Configuration Testing

Section 1.1: Color wheel tests

When: Set *colorWheel* value to false
Then: The color wheel UI element will be removed from the user interface.
Result: Pass

When: Set *colorWheel* value to true
Then: The color wheel UI element will be present in the user interface.
Result: Pass

Section 1.2: Right Panel UI tests

When: Set *radiusSlider* value to true, *windSlider* value to true, *InflateDeflateButton* value to true
Then: The right panel will display all of its UI elements including the *radiusSlider* panel, *windSlider* panel, *inflateDeflateButton* panel, and the switch box panel for switching the wind and radius sliders.
Result: Pass

When: Set *radiusSlider* value to false, *windSlider* value set to true, *inflateDeflateButton* value set to false
Then: The *radiusSlider* UI panel and the switch box for switching to the wind and radius panel will be removed from the user interface. Only the wind slider will be displayed and the *radiusSlider* and *inflateDeflateButton* will not be available.
Result: Pass

When: Set *radiusSlider* value to false, *windSlider* value set to true, *inflateDeflateButton* value set to true
Then: The *radiusSlider* slider bar will be removed from the user interface. The wind slider, switch box for the wind and radius panels, and *inflateDeflateButton* will be available and *radiusSlider* will not be available.
Result: Pass

When: Set *radiusSlider* value to false, *windSlider* value set to false, *inflateDeflateButton* value set to true
Then: The *radiusSlider* bar, *windSlider* bar, and the switch box for swapping the wind and radius panels will be removed from the user interface. Only the *inflateDeflateButton* will be available for use.
Result: Pass

When: Set *radiusSlider* value to true, *windSlider* value set to false, *inflateDeflateButton* value set to true
Then:
Result:

When: Set *radiusSlider* value to true, *windSlider* value set to false, *inflateDeflateButton* value set to true
Then: The *radiusSlider* UI panel will be displayed with the *inflateDeflateButton* on the panel. The switch box for switching between the *windSlider* will disappear and the *windSlider* will not be available.
Result: Pass

When: Set *radiusSlider* value to true, *windSlider* value set to false, *inflateDeflateButton* value to false
Then: The *radiusSlider* UI panel will be displayed and the switch box for switching between *windSlider* and *radiusSlider* will be removed. The *inflateDeflateButton* UI element will be removed from the screen. Only the *radiusSlider* will be available for use on the right panel.
Result: Pass

When: Set *radiusSlider* value to false, *windSlider* value set to false, *inflateDeflateButton* value to false
Then: The right panel will be completely disabled. The *radiusSlider, windSlider,* and *inflateDeflateButton* will not be available in the user interface as well as the switch box for swapping between the wind and radius slider.
Result:

Section 1.3: Left Panel UI tests

When: set *graph* value to true, set *recordButton* to true
Then: graph will be provided for the user as well as the record button to record data on the graph
Result: Pass

When: set *graph* value to false, set *recordButton* to true
Then: graph will be removed from the user interface and the record button will be labeled as an export button to export the data to a CSV
Result: Pass

When: set *graph* value to true, set *recordButton* to false
Then: graph and record button are removed from the user interface
Result: Pass

When: set *graph* value to false, set *recordButton* to false

Then: *graph* and *recordButton* are removed from user interface
Result: Pass

Section 1.4: Radius slider value configuration testing

When: *maxRadius* is set to 500
Then: Radius slider maximum in the user interface is set to default 400
Result: Pass

When: *maxRadius* is set to 300
Then: Radius slider maximum in the user interface is set to 300
Result: Pass

When: *maxRadius* is set to 30*, minRadius* is set to 50
Then: Radius slider maximum in the user interface is set to default 400
Result: Pass

When: *minRadius* is set to 40
Then: Radius slider minimum is set to default 50
Result: Pass

When: *minRadius* is set to 60
Then: Radius slider minimum is set to 60
Result: Pass

When: *minRadius* is set to 410, *maxRadius* is set to 400
Then: Radius slider minimum is set to default 50
Result: Pass

Section 1.5: Wind slider value configuration testing

When: *minWindSpeed* is set to -1
Then: Wind speed slider minimum is set to default 0
Result: Pass

When: *minWindSpeed is set to 4, maxWindSpeed is set to 3*
Then: Wind speed slider minimum is set to default 0 and maximum set to default 5
Result: Pass

When: *minWindSpeed* is set to 3, and *maxWindSpeed* is set to 3
Then: Max wind speed value of slider is set to default 5, and minimum wind speed value of slider is set to 0
Result: Pass

When: *maxWindSpeed* is set to 6

Then: Wind speed slider max is set to 5
Result: Pass

When: *maxWindSpeed* value is set to 4, and *minWindSpeed* value is set to 2
Then: The maximum of the wind speed slider is set 4 and the minimum of the wind speed slider is set to 2
Result: Pass

Section 1.6: Inflate and Deflate buttons

When: *inflateIncrement* value set to 11
Then: Inflate button increases radius by a value of 11
Result: Pass

When: *deflateIncrement* value set to 11
Then: Deflate button decreases radius by a value of 11
Result:

When: *inflateIncrement* value set to 401
Then: Inflate button value set to default 10
Result: Pass

When: *deflateIncrement* value set to 401
Then: Deflate button value set to default 10
Result: Pass

When: *inflateIncrement* value set to 0
Then: Inflate button value set to 10
Result: Pass

When: *deflateIncrement* value set to 0
Then: Deflate button value set to 10
Result: Pass

Section 1.7: Export paths

When: *imageExportPath* is changed from Exports to Exportsimage
Then: Image will be exported to Exportsimage after pressing export button on magnified graph
Result: Pass

When: *csvExportPath* is changed from Exports to Exportcsv
Then: CSV will be exported to Exportcsv after pressing export button on magnified graph
Result: Pass

Section 2: UI Testing

When: Radius slider moved up
Then: Balloon size radius, radius data, surface data, volume data, and force data all increase
Result: Pass

When: Radius slider moved down
Then: Balloon size radius, radius data, surface data, volume data, and force data all decrease
Result: Pass

When: Wind moved up
Then: Wind speed is increased, and more wind is exerted on the balloon
Result: Pass

When: Wind slider moved down
Then: Wind speed is decreased, and less wind is exerted on the balloon
Result: Pass

When: Magnify button is pressed
Then: The graph is increased in size over the UI
Result: Pass

When: Magnify button is pressed after activating larger graph
Then: The graph is decreased in sizer over the UI
Result: Pass

When: Record button is pressed
Then: A point is placed on the graph
Result: Pass

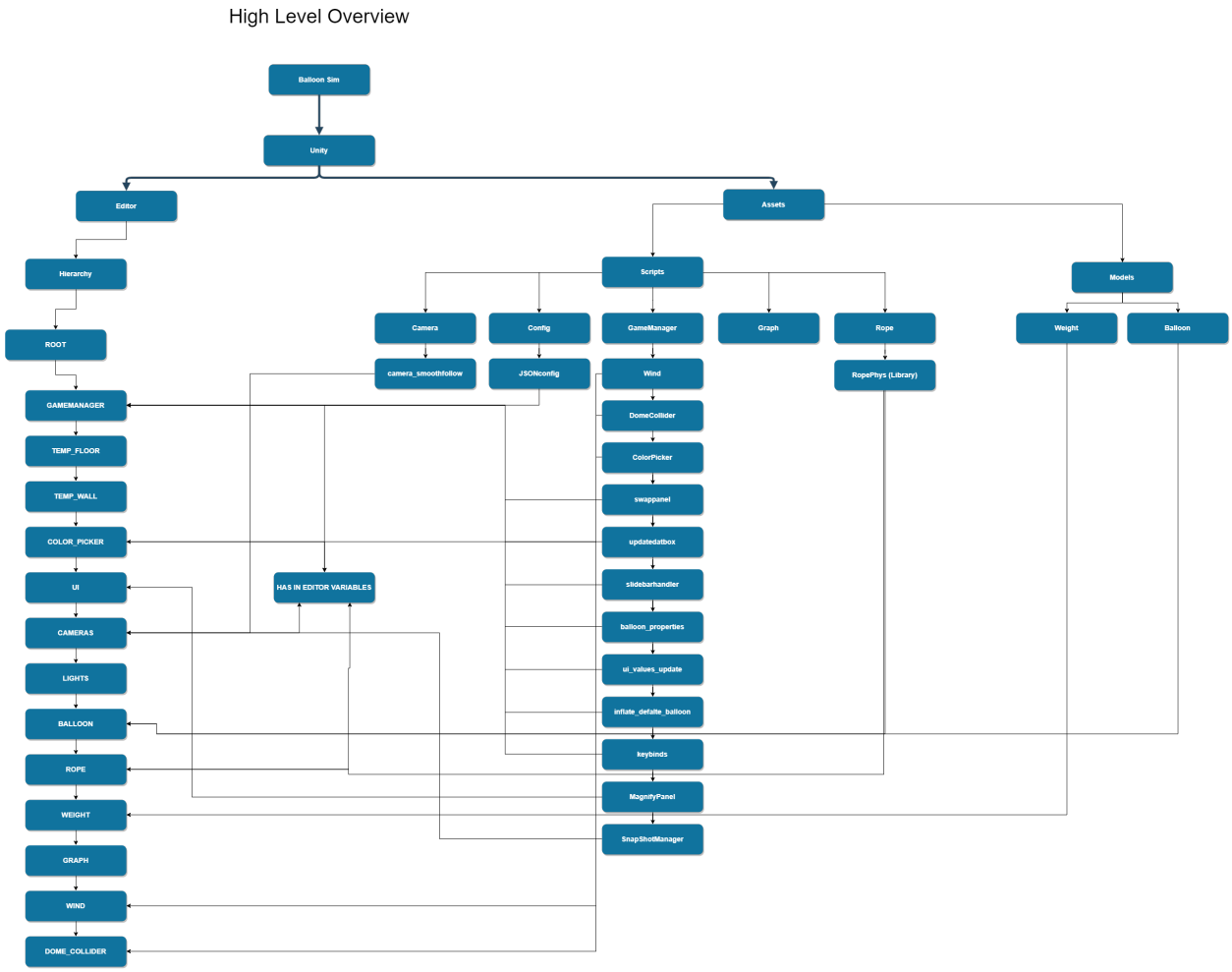
When: Export Button is pressed
Then: Screenshot of the graph is exported to the export link
Result: Pass

When: Modify radius, button is pressed
Then: Radius modification panel is shown on the screen
Result: Pass

When: Modify wind, button is pressed
Then: Wind modification panel is shown on the screen
Result: Pass

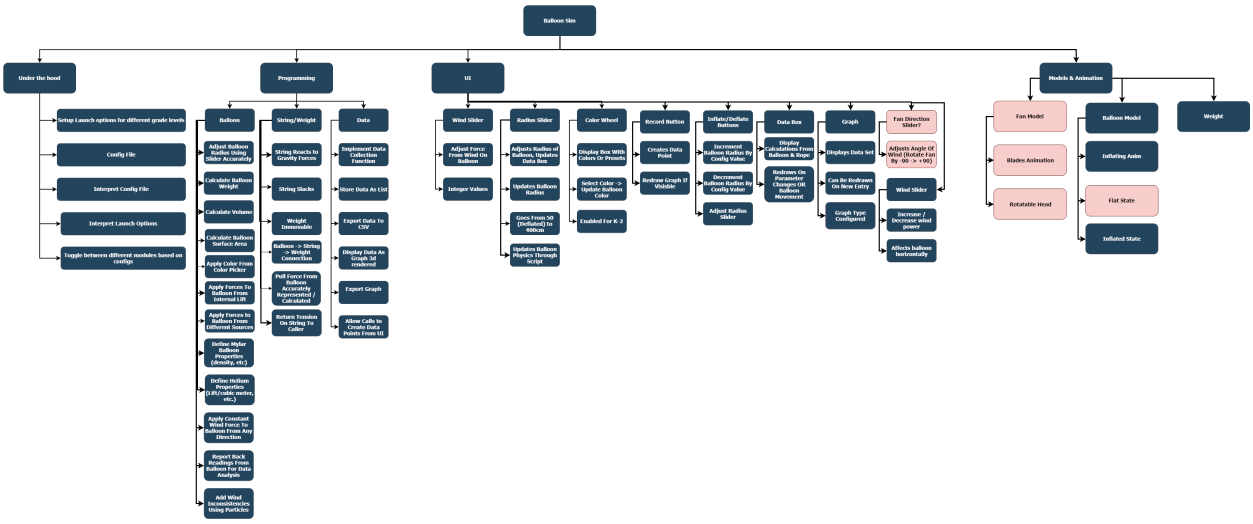
High-Level Organization Diagram

The following diagram depicts what scripts are attached to which objects and which items have in-editor controls which can only be modified through the unity editor.



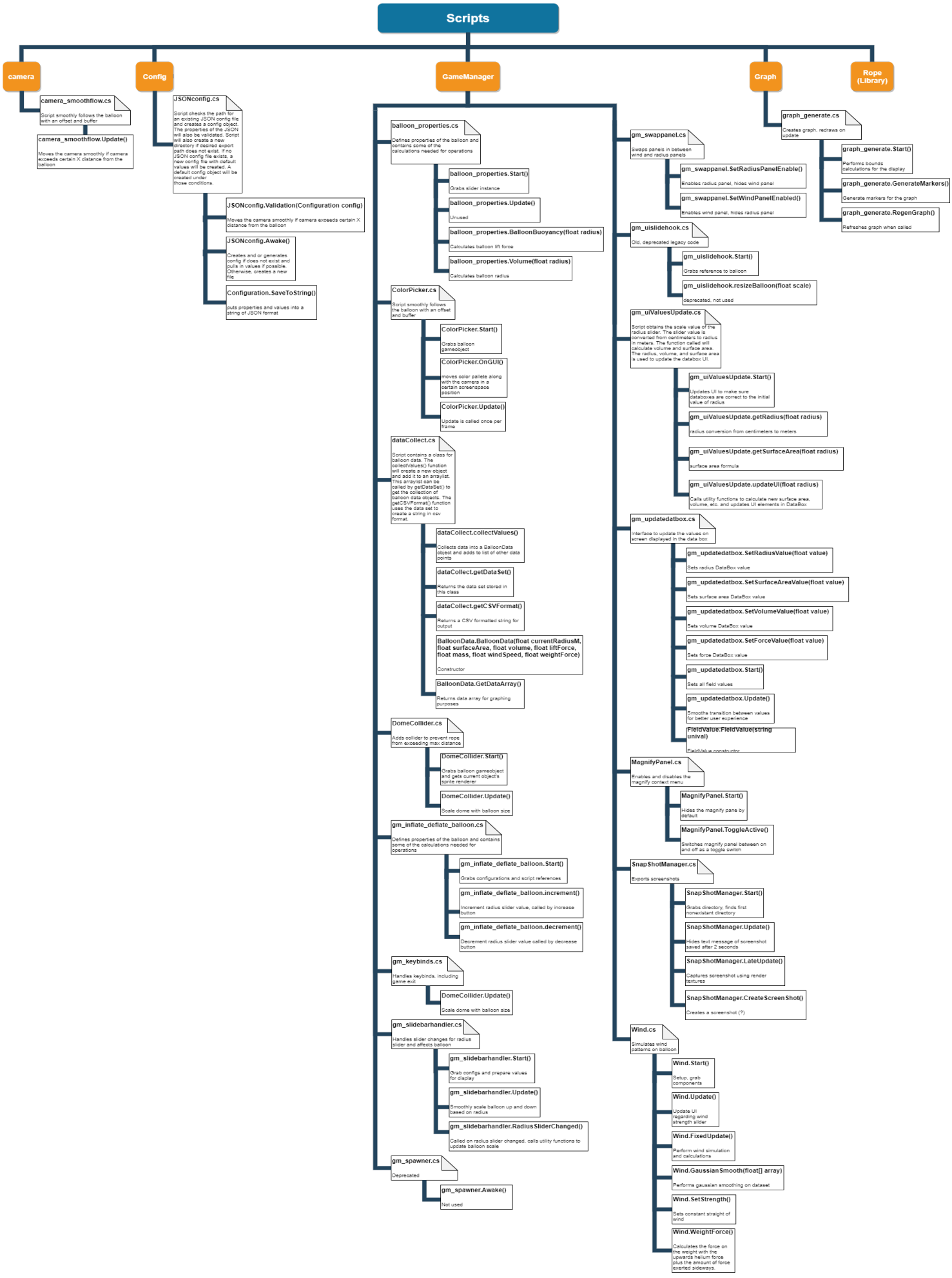
Low-Level Feature Diagram

The following diagram was the original design features diagram. Red items were removed from the final product.



Detailed Script Diagram

The following diagram details the function of each non-deprecated script currently in use


Scripts
camera
camera_smoothflow.cs
Script smoothly follows the balloon with an offset and buffer
camera_smoothflow.Update()
Moves the camera smoothly if camera exceeds certain X distance from the balloon
Config
JSONconfig.cs
Script checks the path for an existing JSON config file and creates a config object. The properties of the JSON will also be validated. Script will also create a new directory if desired export path does not exist. If no JSON config file exists, a new config file with default values will be created. A default config object will be created under those conditions.
JSONconfig.Validation(Configuration config)
Moves the camera smoothly if camera exceeds certain X distance from the balloon
JSONconfig.Awake()
Creates and or generates config if does not exist and pulls in values if possible. Otherwise, creates a new file
Configuration.SaveToString()
puts properties and values into a string of JSON format
GameManager
balloon_properties.cs
Defines properties of the balloon and contains some of the calculations needed for operations
balloon_properties.Start()
Grabs slider instance
balloon_properties.Update()
Unused
balloon_properties.BalloonBuoyancy(float radius)
Calculates balloon lift force
balloon_properties.Volume(float radius)
Calculates balloon radius
ColorPicker.cs
Script smoothly follows the balloon with an offset and buffer
ColorPicker.Start()
Grabs balloon gameobject
ColorPicker.OnGUI()
moves color pallete along with the camera in a certain screenspace position
ColorPicker.Update()
Update is called once per frame
dataCollect.cs
Script contains a class for balloon data. The collectValues() function will create a new object and add it to an arraylist. This arraylist can be called by getDataSet() to get the collection of balloon data objects. The getCSVFormat() function uses the data set to create a string in csv format.
dataCollect.collectValues()
Collects data into a BalloonData object and adds to list of other data points
dataCollect.getDataSet()
Returns the data set stored in this class
dataCollect.getCSVFormat()
Returns a CSV formatted string for output
BalloonData.BalloonData(float currentRadiusM, float surfaceArea, float volume, float liftForce, float mass, float windSpeed, float weightForce)
Constructor
BalloonData.GetDataArray()
Returns data array for graphing purposes
DomeCollider.cs
Adds collider to prevent rope from exceeding max distance
DomeCollider.Start()
Grabs balloon gameobject and gets current object's sprite renderer
DomeCollider.Update()
Scale dome with balloon size
gm_inflate_deflate_balloon.cs
Defines properties of the balloon and contains some of the calculations needed for operations
gm_inflate_deflate_balloon.Start()
Grabs configurations and script references
gm_inflate_deflate_balloon.increment()
increment radius slider value, called by increase button
gm_inflate_deflate_balloon.decrement()
Decrement radius slider value called by decrease button
gm_keybinds.cs
Handles keybinds, including game exit
DomeCollider.Update()
Scale dome with balloon size
gm_slidebarhandler.cs
Handles slider changes for radius slider and affects balloon
gm_slidebarhandler.Start()
Grab configs and prepare values for display
gm_slidebarhandler.Update()
Smoothly scale balloon up and down based on radius
gm_slidebarhandler.RadiusSliderChanged()
Called on radius slider changed, calls utility functions to update balloon scale
gm_spawner.cs
Deprecated
gm_spawner.Awake()
Not used
gm_swappanel.cs
Swaps panels in between wind and radius panels
gm_swappanel.SetRadiusPanelEnable()
Enables radius panel, hides wind panel
gm_swappanel.SetWindPanelEnabled()
Enables wind panel, hides radius panel
gm_uislidehook.cs
Old, deprecated legacy code
gm_uislidehook.Start()
Grabs reference to balloon
gm_uislidehook.resizeBalloon(float scale)
deprecated, not used
gm_uiValuesUpdate.cs
Script obtains the scale value of the radius slider. The slider value is converted from centimeters to radius in meters. The function called will calculate volume and surface area. The radius, volume, and surface area is used to update the databox UI.
gm_uiValuesUpdate.Start()
Updates UI to make sure databoxes are correct to the initial value of radius
gm_uiValuesUpdate.getRadius(float radius)
radius conversion from centimeters to meters
gm_uiValuesUpdate.getSurfaceArea(float radius)
surface area formula
gm_uiValuesUpdate.updateUI(float radius)
Calls utility functions to calculate new surface area, volume, etc. and updates UI elements in DataBox
gm_updatedatbox.cs
Interface to update the values on screen displayed in the data box
gm_updatedatbox.SetRadiusValue(float value)
Sets radius DataBox value
gm_updatedatbox.SetSurfaceAreaValue(float value)
Sets surface area DataBox value
gm_updatedatbox.SetVolumeValue(float value)
Sets volume DataBox value
gm_updatedatbox.SetForceValue(float value)
Sets force DataBox value
gm_updatedatbox.Start()
Sets all field values
gm_updatedatbox.Update()
Smooths transition between values for better user experience
FieldValue.FieldValue(string unival)
FieldValue constructor
MagnifyPanel.cs
Enables and disables the magnify context menu
MagnifyPanel.Start()
Hides the magnify pane by default
MagnifyPanel.ToggleActive()
Switches magnify panel between on and off as a toggle switch
SnapShotManager.cs
Exports screenshots
SnapShotManager.Start()
Grabs directory, finds first nonexistant directory
SnapShotManager.Update()
Hides text message of screenshot saved after 2 seconds
SnapShotManager.LateUpdate()
Captures screenshot using render textures
SnapShotManager.CreateScreenShot()
Creates a screenshot (?)
Wind.cs
Simulates wind patterns on balloon
Wind.Start()
Setup, grab components
Wind.Update()
Update UI regarding wind strength slider
Wind.FixedUpdate()
Perform wind simulation and calculations
Wind.GaussianSmooth(float[] array)
Performs gaussian smoothing on dataset
Wind.SetStrength()
Sets constant straight of wind
Wind.WeightForce()
Calculates the force on the weight with the upwards helium force plus the amount of force exerted sideways.
Graph
graph_generate.cs
Creates graph, redraws on update
graph_generate.Start()
Performs bounds calculations for the display
graph_generate.GenerateMarkers()
Generate markers for the graph
graph_generate.RegenGraph()
Refreshes graph when called
Rope (Library)