Design Document

# 

# 

|  |  |
| --- | --- |
| **Authors:** | Richard Dyer |
|  | Zisis Damianidis |
|  | Tsanko Hadzhiev |
|  |  |
| **Created:** | 02/12/2016 |
| **Last Revision:** | 18/01/2017 |

# 

# 

# 

[**INTRODUCTION**](#_wae4bma9lxxz)[**2**](#_wae4bma9lxxz)

[Purpose of Design Document](#_5nfz6ot00qi8) [2](#_5nfz6ot00qi8)

[**Class Diagram**](#_yzmrcvro88l3)[**3**](#_yzmrcvro88l3)

[**Class Description**](#_jkglxhhjiyej)[**4**](#_jkglxhhjiyej)

[PipelineNetwork](#_dbun9bnyhc3c) [4](#_dbun9bnyhc3c)

[SaveLoad](#_aoe67qruakrg) [5](#_aoe67qruakrg)

[Item (Abstract)](#_6r1zxoy6zboz) [6](#_6r1zxoy6zboz)

[Pipeline](#_l21p956ewuu9) [7](#_l21p956ewuu9)

[Component (Abstract)](#_gfaxmaocvwnn) [8](#_gfaxmaocvwnn)

[Pump](#_7klk5e8ibe72) [8](#_7klk5e8ibe72)

[Merger](#_ebqplslzp4if) [9](#_ebqplslzp4if)

[Splitter](#_3ghloz57f6np) [9](#_3ghloz57f6np)

[Sink](#_jae9cr2u414) [10](#_jae9cr2u414)

[**Sequence diagrams**](#_vnsj6wljqqax)[**11**](#_vnsj6wljqqax)

[Save As](#_s7qxpvs7t0ow) [11](#_s7qxpvs7t0ow)

[Add Component](#_p3axltb3e4wl) [12](#_p3axltb3e4wl)

[Change Flow](#_zhc9xa6mvsp) [13](#_zhc9xa6mvsp)

# 

# 

# INTRODUCTION

## Purpose of Design Document

The Design Documents describes and shows the details of the classes, methods and variables involved in building the application. The Class Description provides a description for all the classes. The Method Description clarifies some of the methods. In the interaction section some of the more abstract methods are shown how they will interact with each other.

# 

# 

# Class Diagram

# Class Description

|  |  |  |  |
| --- | --- | --- | --- |
| Method/Variable | Name | Description | Returns/Type |
|  | PipelineNetwork | | |
|  | PipelineNetwork is the main class, manager class which is in charge of Adding/Removing components,checking for placement issues, and telling the components how and where to draw themselves. | | |
| Variable | * SelectedItem | Represents the Item the user has selected in the Pipeline Network layout, or in the Component selection panel. | Object(Item) |
| Variable | * SelectedItem2 | Represents the second selected Item, used for creating a pipeline. | Object(Item) |
| Variable | * Items | A collection of all the items placed in the Pipeline Network Layout. | List<Object(Item)> |
| Variable | + PipelineName | The name of the Network. Used for saving and loading. | string |
| Variable | -ComponentSize | A number that represents component size. | int |
| Variable | -pipelineWidth | A number that stores the width of a pipeline | int |
| Variable | +SelectedItem1ChangedHandler(Item selectedItem); | Whenever an item is selected this delegate raises SelectedItem1Event;event. | Delegate |
| Variable | +SelectedItem1Event; | An event which is raised , whenever a component is clicked. | Event |
| Variable | +DrawItemsHandler(); | A delegate which is invoked whenever a component is added,removed, replaced or changed | Delegate |
| Variable | +DrawItemsEvent |  | Event |
| Variable | +NetWorkErrorHandler(string error); |  | Delegate |
| Variable | +NetworkErrorEvent; | A delegate for showing an error message, whenever an action is not possible. | Event |
| Variable | * fileHandler | Object used for handling saving and loading. | Object SaveLoad |
| Method | +disselect() |  |  |
| Method | +NetworkClicked(Point positiob) | This methods raises SelectedItem1Event whenever the user clicks on an item. |  |
| Method | +NetworkDoubleClicked(string Component, Point Position) | Adds a new components to the pipeline layout whenever user double clicks it |  |
| Method | +AddComponent(Item i,Point pos):bool | This methods adds a new component(item) to the list of “items”. And will make the item draw itself. Will call “canPlaceItem” before executing. | True if component is added successfully or false if failed |
| Method | - CheckForLoop(Point pos):bool | Checks if the network would loop if pipeline was placed | Returns true if network would loop. |
| Method | Remove(Item i ): bool | Goes through list of items and removes the selected item.. | Returns true or false, based on the result. If deleting was successful returns true else return false |
| Method | - calculateNetwork() | Calculates and orders items to either draw themselves a specific color, or set values based on connections. |  |
| Method | - canPlaceItem(Point pos):bool | Calculates if a component can be placed in the area. | Returns true if component can be placed, returns false if area is already occupied. |
| Method | +AddPipeline(Item i, Item i2):bool | Checks if both items can have pipelines and then uses “PathClear” and “checkForLoop” then creates a pipeline connected to the two items and tells the pipeline to draw itself. | Returns true if pipeline was added. Returns false if items already has pipelines or if a loop would have occurred or if path was not clear. |
| Method | - PathClear(Item i, Item i2):bool | Check if a pipeline were drawn in between the two items if it intersects with a component. | Returns true if no intersection would occur. |
| Method | + getItem(Point pos) | Gets the item with the sought after position. | Returns the item with the position, returns null if no item was found. |
| Method | +ReplaceComponent(string Component, Component replace) | Replaces the selected component with the selected new one | bool |
| Method | -CheckCanConnect(Component c1Output, Component c2Input) | Returns true if connection between the given 2 component is possible. | bool |
| Method | -ComponentInPosition(Component c , Point pos) | Checks if the given position is suitable for the component | bool |
| Method | PipelineInPosition(Pipeline p ,Point pos) |  | bool |
| Method | -PipelineInComponentPosition(Pipeline p, Point pos) |  |  |
| Method | -getDistance(Point A, Point B, Point C, out double ab, out double ac , out double bc) | This method is used to calculate the Area around a component |  |
| Method | -getArea(Point A, Point B, Point C) | Returns the area around the 3 given points | double |
| Method | -bool PipelinesIntersect(Component ab, Component ba, Component cd , Component dc) | Checks if the intercept point x and y is indeed between the two lines. | bool |
| Method | -setDrawDirection(Component component) | Assigns component direction to the next component |  |
| Method | +ComponentIsSelected() | Returns true if selected item is not empty | bool |
| Method | getItems() | Returns list of all items | List<item> |
|  | SaveLoad | | |
|  | The purpose of this class is to Save/Load a pipeline network to/from a binary file. | | |
| Method | +Save(string file,List<item> items):Bool | Uses “ItemsToBytes” then saves the Items to a file. | bool |
| Method | +Load(string filename): List<Item> | Uses “BytesToItems” then returns a List of items. | Returns a list of litems. |
| Method | - ItemsToBytes(List<item> items) : byte[] | Turns the list of items into bytes. | Returns null if unsuccessful.Othewise returns a byte array. |
| Method | - BytesToItems(byte[] bytes) : List<items> | Turns bytes into a list of items | Return null if unsuccessful.  Otherwise returns a list of items. |
|  | Item (Abstract) | | |
|  | Item is the base class for all components and pipelines. | | |
| Variable | Flow | Each Item has a flow | int |
| Variable | IsConnectedToPump | If this item or one of its connecting items are connected to a pump | bool |
| Variable | IsConnectedToSink | If this item or one of its connected items are connected to sink. | bool |
| Method | <<abstract>> SetOutputFlow( ) : bool | Each item needs to be able to set the flow of the connected component. | Returns false if no output item or if output item is a sink. |
| Method | + GetFlow() :double | Gets the items current flow. | Returns the flow or -1 if no it has no flow. |
| Method | + SetFlow() :double | Sets the item flow | int |
| Method | <<abstract>>Component getNextComponent() | Returns the next component | Object Component |
| Method | <<abstract>>Pipeline getNextPipeline() |  | Pipeline Object |
|  | Pipeline | | |
|  | Pipeline inherits from Item class. Pipeline has extra parameters such as an Input and OutPut Component. | | |
| Variable | SafetyLimit | The safety limit of the pipeline if the pipeline flow is over the safety limit the pipeline will know to draw itself red. | double |
| Variable | input | The component connected to the pipeline’s input. | Object Component |
| Variable | output | The component connected to the pipeline’s output. | Object Component |
| Constructor | +Pipeline(Component comp, Component comp2, SafetyLimit Double) | This is the constructor for Pipeline class , it needs to know which two components it will be placed. And the Safety limit to be able to create a Pipeline object. |  |
| Method | +SetSafetyLimit(double safetyLimit):void | Assigns the given value to the safety limit. | void |
| Method | + CheckPumpSinkConnection():void | Checks if the pipelines input or output is a sink or pump if yes, then it will change all connected item’s “IsConnectedToPump”/”IsConnectToSink” variables to true. | void |
| Method | + getSafetyLimit( ) : double | Returns the current safety limit of the pipeline. | double |
|  | Component (Abstract) | | |
|  | A component is the base class for all components. It also inherits from Item but has additional attributes such as a point. | | |
| Variable | + Output | The Pipeline in the components output. | Object(Pipeline) |
| Variable | - Position | The location of the component. | Point |
| Constructor | +Component(Point p) | The location of the component is needed to set a component | void |
| Method | +GetPosition():Point | Returns the location of the selected component | Point |
| Method | +AddOutput(Pipeline pipe): bool | Set the Component’s output pipeline. | True if successful, false if it failed or component already has a pipeline attached. |
|  | Pump | | |
|  | Stores information for pump components. Pump has capacity as extra parameter. Pump is the only component where a flow can be directly set. And is the origin of a networks flow. | | |
| Variable | - Capacity | The maximum flow the pump can output. |  |
| Constructor | +Pump(Point p, Capacity c, Flow double) | A pump needs a point, capacity, and flow to be created. | void |
| Method | +ChangeCapacity(double cap):void | Change the capacity of the pump. | void |
| Method | +ChangeFlow(double flow):void | Change the flow produced by the pump and change the flow of all connected items. |  |
|  | Merger | | |
|  | Component merging two pipelines into one. | | |
| Variable | + InputA | The first input of the merger. | Object (Pipeline) |
| Variable | + InputB | The second input of the merger | Object(Pipeline) |
| Variable | - flow2 | If the first flow is filled when the second input changes this component's flow it will be set instead of the first. | double |
| Constructor | +Merger(Point p) | Pump constructor, similar to the base class. |  |
| Method | +AddIntput(Pipeline pipe):void | This method assigns input to the selected component. | void |
| Method | +AddIntputB(Pipeline pipe):void | Assigns inputB to this component | void |
|  | Splitter | | |
|  | Splitter splits one pipeline into two (can also adjust the ratio of the split) | | |
| Variable | -InputA | The input of the splitter | Object (Pipeline) |
| Variable | -OutputB | This is the second output of the splitter | Object (Pipeline) |
| Variable | -adjustPercenatage | This variable adjust the flow of output pipelines. | int |
| Constructor | +Splitter(Point p, adjustPer int) | Splitter constructor is similar to the base constructor , the only difference is that Splitter has adjustPercentage variable instead of flow. |  |
| Method | +SetAdjustment(int percen) | Sets the adjustmentPercentage value to the given value | void |
| Method | +AddOutputB(Pipeline pipe) | Set the second Splitter’s output pipeline. | bool |
|  | Sink | | |
|  | Sink is the last component of a pipeline network. | | |
| Constructor | +Sink(Point p ) | Constructor only has location , no flow. |  |

# 

# 

# Sequence diagrams

## Save As

Save.png

## Add Component

AddNew.png

## Change Flow

Change Flow.png