Teaching Interactive Computer Science

Group 7

Alexander Kjellén
Björn Delin
Erik Skogby
Jan-Erik Bredahl
Joakim Israelsson

Section 5.5 – 5.6

Detailed Design

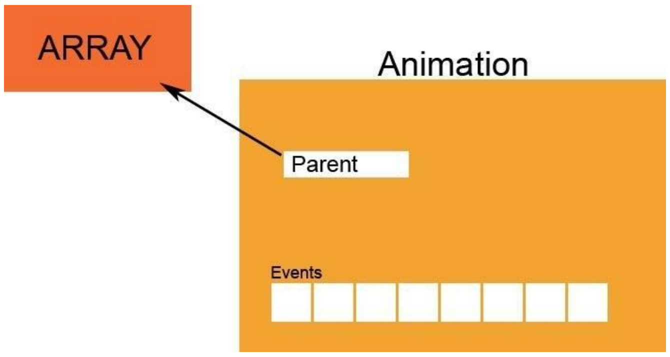
Data structures

We have two internal data structures in our system, they are named: Animation, and Frame.

Animation

This data structure holds a list of animation steps as well as what type of data structure it represents. This structure also has a pointer pointing back at the structure it was created from. The available animations steps are:

1. CreateNode
2. DeleteNode
3. SwapNode
4. CreatePointer
5. DeletePointer
6. SwapPointer
7. MoveNode



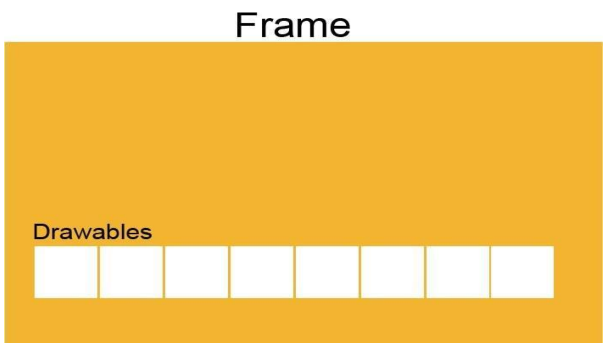
It is created in the Structure class and thereafter used in the AnimationController class.

Every node keeps track of at least three things: id, value and type. Some might have to keep track of its parent.

Frame

The frame structure represents a picture of the animation in a specific condition. It contains a list of Drawable objects that knows how they should be drawn to the screen. This Drawable object contains all the necessary information to draw the object, such as coordinates, object type, etc. Examples of Drawable objects are:

1. Pointers
2. ArrayNodes
3. LinkedListNodes
4. SortedBinaryTreeNodes
5. TemporaryNodes
6. Header



This structure is used by the Animation window when it draws the animation on the screen. It is created by the FrameCreator class which sends a certain amount of Frames per second.

Methods

This list of methods contains all major methods. Various GUI methods, smaller helper methods and private methods are not included.

MainWindow

Int main(String[] args)

Program starting point. It sets up all the GUI classes, creates an Animation Controller and a DataStructures object.
Parameters: Not used
Return value: Zero for successful execution, otherwise non-zero.
Pre-conditions: Nothing is set up at all.
Post-Conditions: The program is up and running successfully or a non-zero integer is returned from the function.
Called by: Java Virtual Machine
Calls: DataStructures.getAvailableStructures(),DataStructures.getAvailableFunctions()

DataStructures

Void createStructure(String structure)

Creates a structure. It replaces the old structure if necessary.
Parameters: String structure tells the method which structure to create.
Return value: none
Pre-conditions: none
Post-conditions: a data structure is loaded and filled with numbers, if a old one existed, it was replaced.
Called by: Java Swing GUI callbacks
Calls: Structure’s constructor.

Animation runFunction(String function)

This method is just a wrapper for the equivalent method in the current structure.
Parameters: A string specifying the desired function.
Return value: An animation describing the method execution, or null if an error occurred.
Pre-conditions: There must be a current structure loaded.
Post-conditions: See the structure method.
Called by: Java Swing GUI callbacks.
Calls: Equivalent structure method.

String[] getAvailableFunctions(void)

This method is just a wrapper for the equivalent method in the current structure.
Parameters: none
Return value: A list of the available functions in String form, nothing is return if a structure is not loaded.
Pre-conditions: none
Post-conditions: The list of strings is returned.

Called by: Java Swing GUI callbacks and the main method
Calls: Structures.getAvailableFunctions()

String[] getAvailableStructures(void)

A complete set of available Structures is returned.
Parameters: none
Return value: A list of the available structures in String form.
Pre-conditions: none
Post-conditions: The list of strings is returned.
Called by: Java Swing GUI callbacks and the main method
Calls: none

Structure

String[] getAvailableFunctions(void)

Depending on what structure is currently loaded, a complete set of available functions is returned.
Parameters: none
Return value: A list of the available functions in String form, nothing is return if a structure is not loaded.
Pre-conditions: none
Post-conditions: The list of strings is returned.
Called by: DataStructures.getAvailableFunctions()
Calls: none

Animation runFunction(String function)

This method executes a function and returns an animation describing the function.
Parameters: A string specifying the desired function.
Return value: An animation describing the method execution, or null if an error occurred.
Pre-conditions: The function specified by the string must be available.
Post-conditions: Same as before. The structure isn't actually changed before the animation is drawn.
Called by: DataStructures.runFunction()
Calls: none

AnimationController

Void tick()

This method tells the FrameCreator what is changed during a step in the animation.
Parameters: none
Return value: none
Pre-conditions: That an animation is running.
Post-conditions: The animation step has been sent to the FrameCreator.
Called by: Java timer object
Calls: FrameCreator.createFrame(Animation,int)

Void updateGUI()

This method tells the MainWindow what buttons the user can press.
Pre-conditions: none

Post-conditions: The appropriate AnimationControll buttons gets updated.
Called by: Java timer object
Calls: none

FrameCreator

Void createFrame(Animation,int)

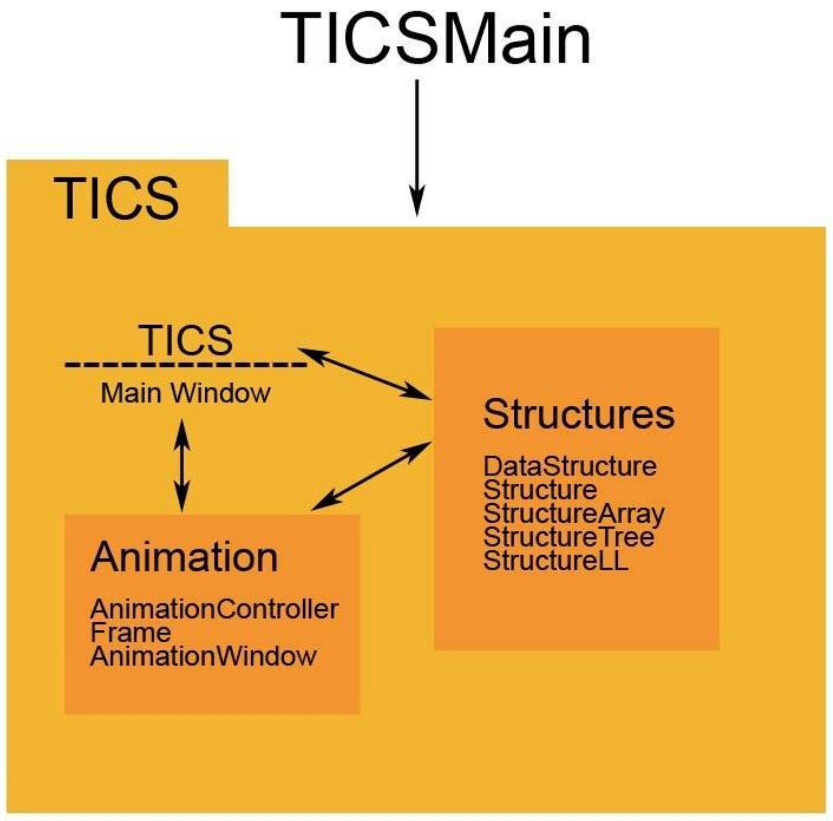
This method takes an Animation and an int and creates a number of Frames which it sends to the AnimationWindow.
Parameters: Animation and an integer.
Return value: none
Pre-conditions:none
Post-conditions: The Frames are sent to the AnimationWindow.
Called by: AnimationController.tick()
Calls: AnimationWindow.draw()

AnimationWindow

void Draw(Frame arg)

Draws a frame in the animation area.
Parameters: a frame with all objects' positions.
Return value: none.
Pre-conditions: none.
Post-conditions: the frame has been drawn.
Called by: FrameCreator.createFrame()
Calls: none

Package Diagram


TICSMain
TICS
TICS
Main Window
Structures
DataStructure
Structure
StructureArray
StructureTree
StructureLL
Animation
AnimationController
Frame
AnimationWindow