

# ECE231 Final Project

Acequia Manager: Water Resource Simulator

# Acequia: a Brief Context

- Water resources in New Mexico are connected by a large network of canals that deliver water to communities
- Within New Mexico, these canals are commonly known as Acequias
  - Their common purpose has been to deliver water for irrigation to crops and ranching
- In the case of this simulator, Acequias serve as a metaphor for water transfer that is NOT exclusive to irrigation.
  - Rather, these “canals” consider water movement across the state

# Why Acequia Manager Simulation?

- This exercise is intended to build on using programming languages to solve real-life problems
  - We hope to strengthen a student's problem-solving skills using C++
  - We hope to enable this exercise in an environment that is analogous to coding interviews

# How the Simulator Works

The Simulator populates the vectors for Regions, Water Sources, and Canals

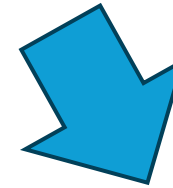
Each class object will be assigned random values

The values of each region object will be displayed on the screen as such:

Region: Name    Water Level: \_\_\_\_\_  
Flooded: \_\_\_\_\_  
Drought: \_\_\_\_\_

North Region:  
WaterSources

- Rio Grande River
  - ABQ Aquifer
  - Pecos River
- WaterLevel
- 4.301 (acre-feet)
  - Flooded? No
  - Drought? No



Canal A:

- Tapped from Rio Grande
- Source Region: North
- Destination Region: South

South Region:  
waterSources

- Elephant Butte Dam
  - Rio Grande River
- WaterLevel
- 1.301 (acre-feet)
  - Flooded? No
  - Drought? Yes

# How the Simulator Works (cont.)

- Based on these values, the student is provided getter functions to access the members of: region, waterSources, and canals
  - Students will facilitate the transfer of water resources from one region to another using the canals objects in the solve problem function
- Students are given a certain number of hours (simulation time units) that they can implement actions to address distribution of water resources across all regions
  - In those actions, the student may open and close canals, and adjust flow rates

# How the Simulator Works (cont.)

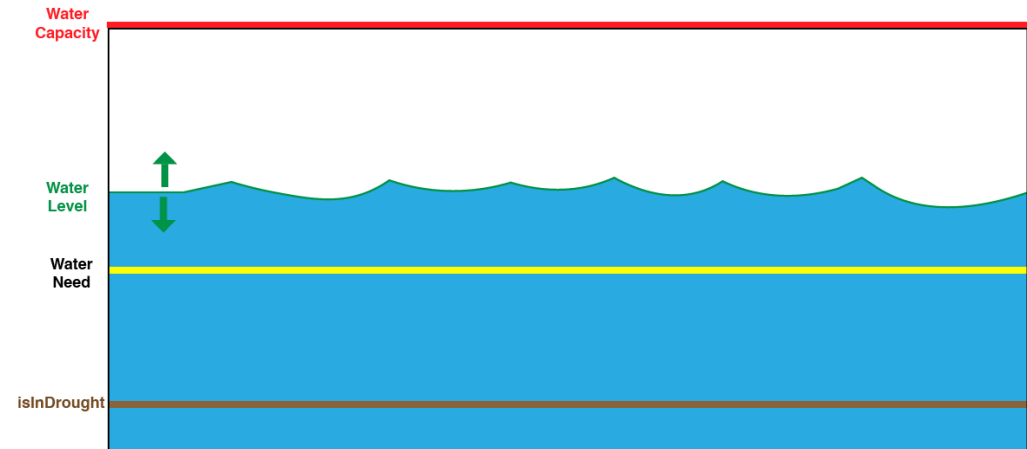
- Once the solve problem function has been added, the simulator evaluates the final state of each region and scores one point based on satisfying the following criteria:
  - The waterLevel of the region is above or equal to the waterNeed of the region
  - The region is NOT flooded. That is indicated by the Boolean, isFlooded
  - The region is NOT in Drought. That is indicated by the Boolean, isInDrought
- A total score is computed and reported with the current leaderboard presented

# Simulator Description

- The simulator handles
  - populating each vector class with Random Parameters and Constraints
  - displaying the information about the particular problem
  - The progress of the simulator (hours provided to the student)
    - And the exchanges in water resources between regions
  - evaluating the final states of each region and administering a score after the student's solveProblems function has been implemented
  - Displaying the Leaderboard for the simulator

# Region Class Members

- **WaterLevel**(double): Shows the current water level of the region
- **WaterNeed**(double): shows the amount of water needed to support the communities of the region
- **waterCapacity**(double): shows the amount of water the region can hold before flooding
- **isInDrought**(bool): Shows the status of the region when water levels have reached drought levels
- **isFlooded**(bool): Shows the status of whether the water levels have reached flooding conditions

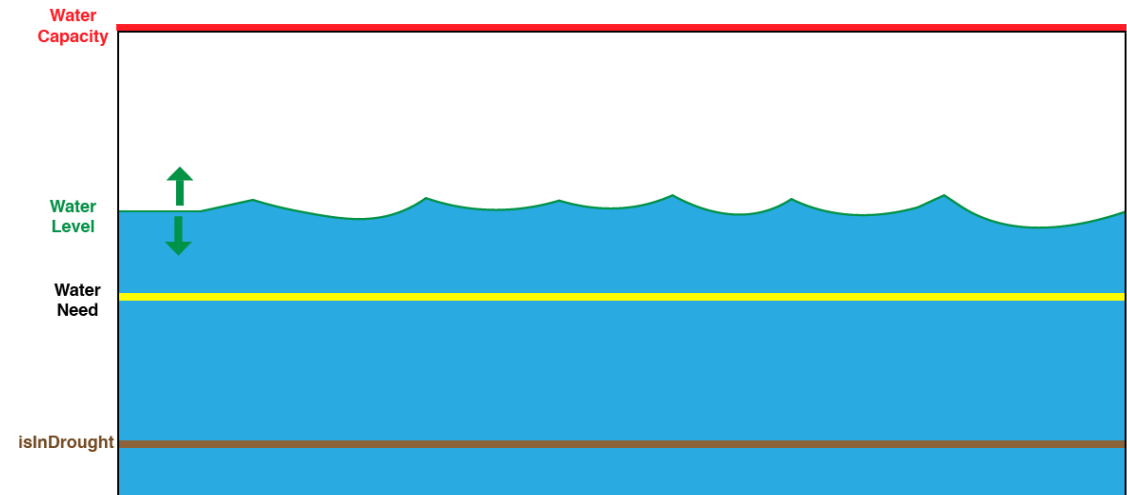




# Region Class

## Members cont.

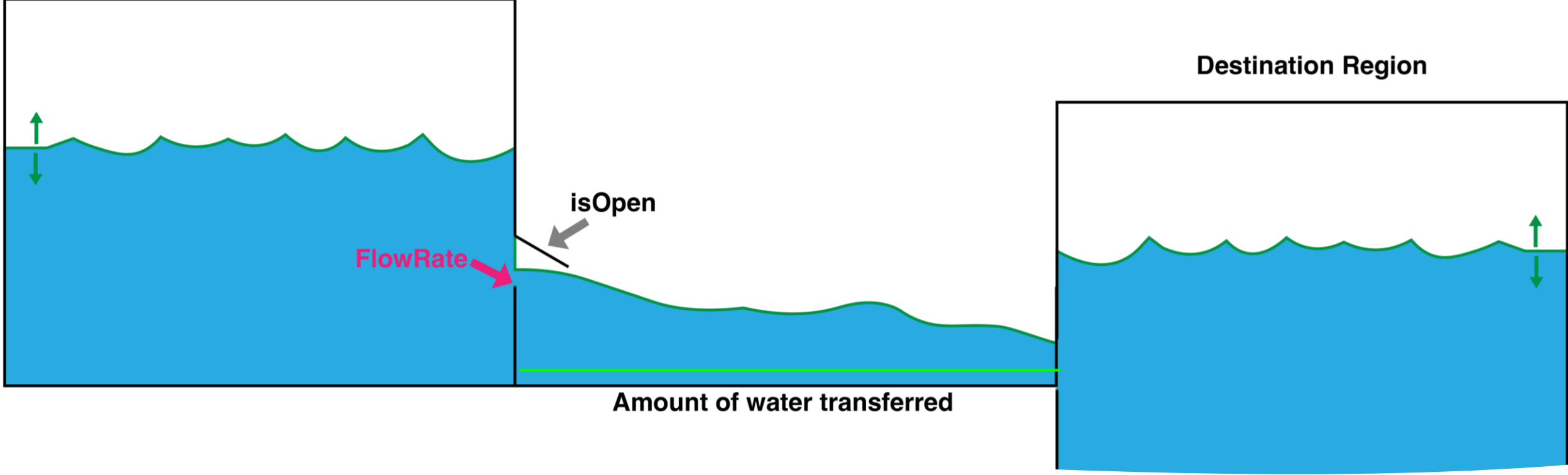
- **Vector of WaterSources**  
(vector<WaterSource \*>)
  - Rivers, dams and aquifers that supply water to that region





## WaterSource Class Members

- Name
- **WaterLevel** (double)
- **WaterSourceType**
  - River
  - Dam
  - Underground
- **Canals** (std::vector<Canals\*>)



## Canal Class Members

- Name
- sourceRegion(Region \*)
- DestinationRegion (Region \*)
- waterSource (WaterSource \*)
- flowrate (double)
- isOpen (bool)

# SolveProblems Function

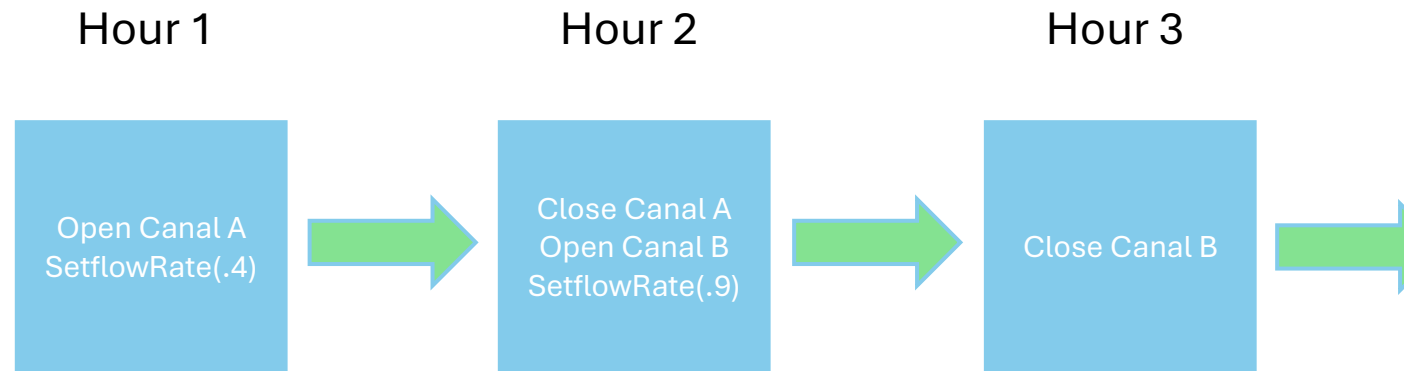
- The solveProblems function can be found in the StudentSolution.cpp file
- In this function, the student will code the logic for how to solve the distribution of water resources
- Student can move water by opening/closing canals and setting flow rates (0 to 1 gal/s)

```
void solveProblems(AcequiaManager& manager)
{
    auto canals = manager.getCanals();
    while(!manager.isSolved && manager.hour!=manager.SimulationMax)
    {
        //Students will implement this function to solve the problems
        //Example: Adjust canal flow rates and directions
        if(manager.hour==0)
        {
            canals[0]->setFlowRate(1);
            canals[0]->toggleOpen(true);
        }
        else if(manager.hour==1)
        {
            canals[1]->setFlowRate(0.5);
            canals[1]->toggleOpen(true);
        }
        else if(manager.hour==82)
        {
            canals[0]->toggleOpen(false);
            canals[1]->toggleOpen(false);
        }
        //student may add any necessary functions or check on the progress of each region as the simulation moves forward.
        //The manager takes care of updating the waterLevels of each region and waterSource while the student is just expected
        //to solve how to address the state of each region

        manager.nexthour();
    }
}
```







# .nexthour()

- This function handles the progress of the simulator and how many actions a student can initiate during that hour and how that changes the water levels of the different regions



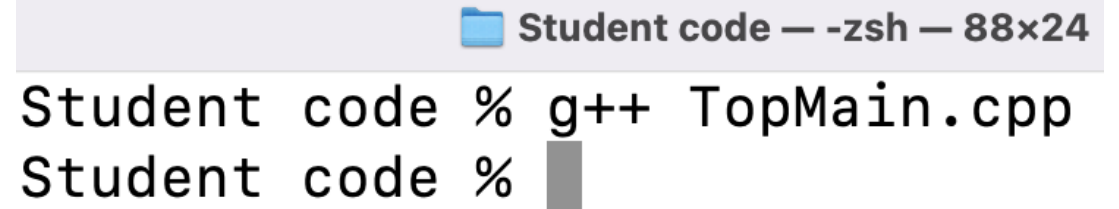
# Getting Started

- Download all files:
  - TopMain.cpp
  - AcequiaManager.cpp
  - StudentSolution.cpp
  - SimulatorMain.cpp
  - AcequiaManager.h

< > Student code	
Name	
	Acequia Manager Guide.docx
	acequia_manager.h
	AcequiaManager.cpp
	<b>SimulatorMain.cpp</b>
	StudentSolution.cpp
	TopMain.cpp

# Getting Started (cont.)

- Compile TopMain.cpp
- Execute the file
  - Ex: ./a.out



```
Student code % g++ TopMain.cpp
Student code %
```

# Running TopMain.cpp

- When running the TopMain.cpp, the user will receive:
  - The initial conditions of each region
  - The simulation time they have to solve the water management
- Lastly, they will be prompted to press Y when ready to test their solveProblem Function
  - Press when ready to activate Simulator

Current State of the Regions:

-----  
Region: North, Water Level: 75, Water Need: 79, Water Capacity: 101  
Region: South, Water Level: 4, Water Need: 68, Water Capacity: 156  
Region: East, Water Level: 42, Water Need: 95, Water Capacity: 150  
-----

Please write your solution in the StudentSolution.cpp file.  
Your code must solve each region's water needs within the following simulation time: 74  
When you have saved your code and ready to run the simulation, you may press Y to run.  
Press Y to test your solveProblems function.





# Before Pressing Y

- Prior to activating the simulator,
  - The user will enter instructions on the solveProblem function based on the parameters provided
- Save the modified StudentSolution.cpp file
- Press Y to activate the simulator

```
void solveProblems(AcequiaManager& manager)
{
    auto canals = manager.getCanals();
    while(!manager.isSolved && manager.hour!=manager.SimulationMax)
    {
        //Students will implement this function to solve the problems
        //Example: Adjust canal flow rates and directions
        if(manager.hour==0)
        {
            canals[0]->setFlowRate(1);
            canals[0]->toggleOpen(true);
        }
        else if(manager.hour==1)
        {
            canals[1]->setFlowRate(0.5);
            canals[1]->toggleOpen(true);
        }
        else if(manager.hour==82)
        {
            canals[0]->toggleOpen(false);
            canals[1]->toggleOpen(false);
        }
        //student may add any necessary functions or check on the progress of each region as the simulation moves forward.
        //The manager takes care of updating the waterLevels of each region and waterSource while the student is just expected
        //to solve how to address the state of each region

        manager.nexthour();
    }
}
```

# Activating the Simulator

- When the simulator is launched,
  - The user will be provided the final state of each region.
    - Water level, water need, if the region is flooded, or in drought
  - They will be notified if their solution solved the problem prior to the max time
    - If so, their solved time will be reported
  - They will receive a score

Press Y to test your solveProblems function.

y

Size of Region :3

Current State:

-----

Region: North, Water Level: 75, Water Need: 79, Flooded: No, Drought: No

Region: South, Water Level: 4, Water Need: 68, Flooded: No, Drought: Yes

Region: East, Water Level: 42, Water Need: 95, Flooded: No, Drought: No

-----

Current State:

-----

Region: North, Water Level: 0, Water Need: 79, Flooded: No, Drought: Yes

Region: South, Water Level: 139, Water Need: 68, Flooded: No, Drought: No

Region: East, Water Level: 150, Water Need: 95, Flooded: Yes, Drought: No

-----

Not all regions were solved in time.

-----

-----

Leaderboard:

StudentSolution:-89

# Scoring Criteria

## Earned Points:

- +10
  - For each region that satisfies:
    - Water level above water need
    - Is not flooded
    - Is not in drought
- +50
  - If solveProblem() function solves each region before max simulation time

## Penalties:

- -1
  - For each hour a region is in a flooded state
- -1
  - For each hour a region is in a drought state