



Introduction to Programming through C++
Professor Abhiram G. Ranade
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Lecture 26 Part 2 – A graphical editor and solver for circuits

(Refer Slide Time: 0:23)

What we discussed

- CAD, Computer aided design is an important application of computers
- Computer based design programs often use graphical user interfaces
- We will design a baby CAD program for designing and solving resistive circuits.

Next: Demo of the program





Demo

- circuit.cpp

Observe:

- Buttons
- How to add nodes, conductances, current sources
- Solver output

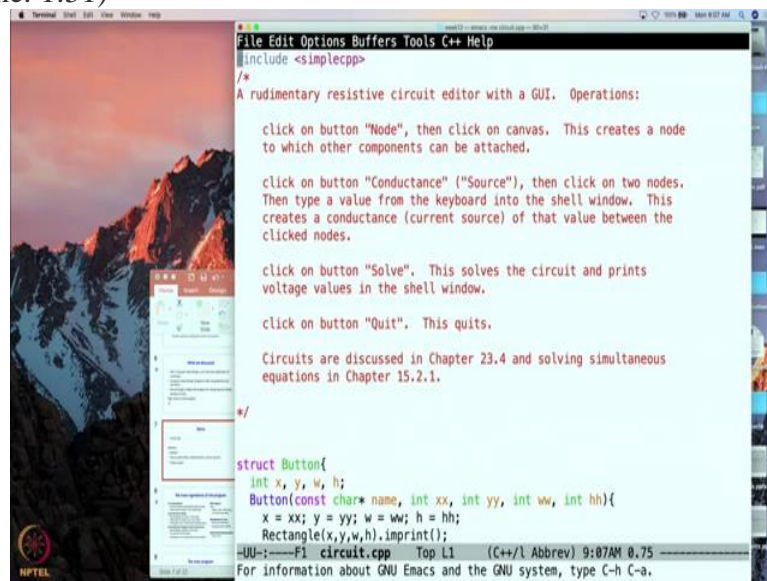


Welcome back. In the previous segment we talked about computer aided design in general and graphical user interfaces. And what we want to do in this lecture which is designing a computer aided design program for solving, for designing and solving resistive circuits. So we will begin this segment with a demo of the program. So the program is in the file circuit.cpp which I will show you but I will, we will discuss the program a little bit more in detail later on. What I want you to observe when we run this program is that there are buttons

in it and then these buttons enable you to add nodes, conductances, current sources. And then the buttons also allow you to solve. And I would like you to observe the solved output.

So, I am going to talk about conductances over here. So conductance, the conductance of an element is the inverse of its resistance. So, these are just these are just two terms used to denote the parameter of a circuit component. So, it makes more sense to talk in terms of the conductance rather than the resistance and that is why we will be using that term. So, let us do the demo.

(Refer Slide Time: 1:51)



```
File Edit Options Buffers Tools C++ Help
#include <simplecpp>
/*
A rudimentary resistive circuit editor with a GUI. Operations:

    click on button "Node", then click on canvas. This creates a node
    to which other components can be attached.

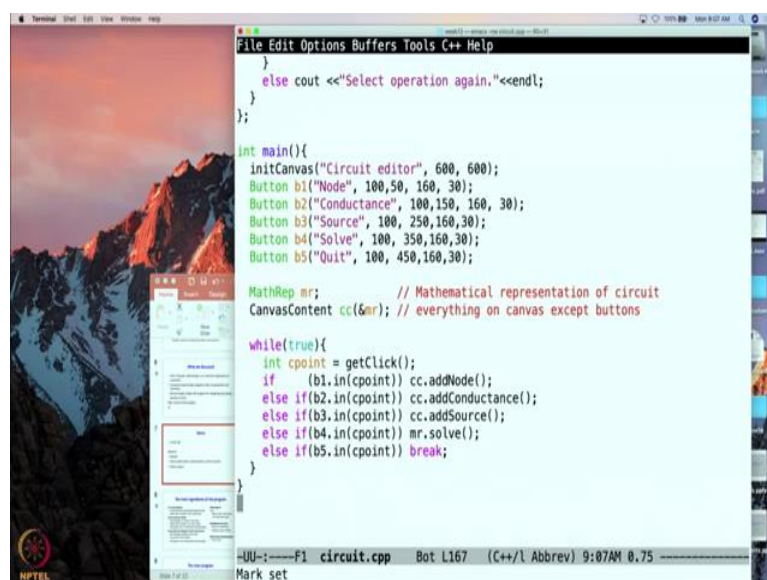
    click on button "Conductance" ("Source"), then click on two nodes.
    Then type a value from the keyboard into the shell window. This
    creates a conductance (current source) of that value between the
    clicked nodes.

    click on button "Solve". This solves the circuit and prints
    voltage values in the shell window.

    click on button "Quit". This quits.

Circuits are discussed in Chapter 23.4 and solving simultaneous
equations in Chapter 15.2.1.
*/

struct Button{
    int x, y, w, h;
    Button(const char* name, int xx, int yy, int ww, int hh){
        x = xx; y = yy; w = ww; h = hh;
        Rectangle(x,y,w,h).Imprint();
    }
};
```

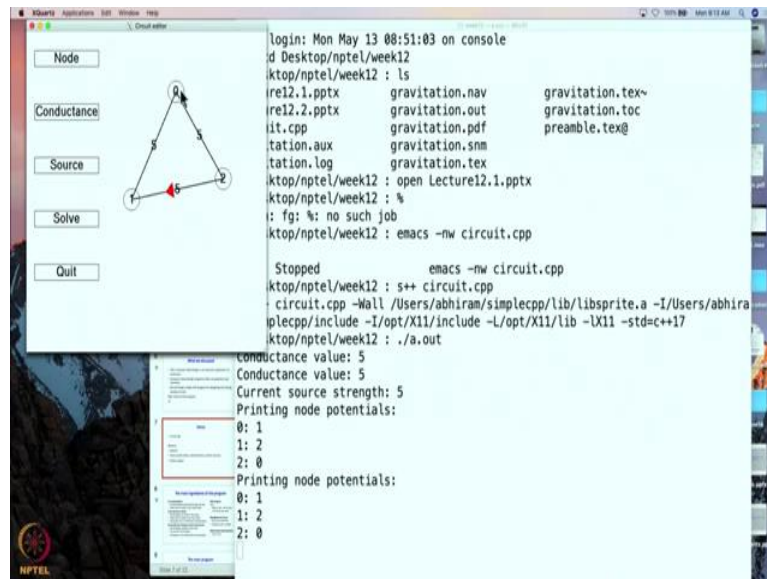


```
    }
    else cout <<"Select operation again."<<endl;
};

int main(){
    initCanvas("Circuit editor", 600, 600);
    Button b1("Node", 100,50, 160, 30);
    Button b2("Conductance", 100,150, 160, 30);
    Button b3("Source", 100, 250,160,30);
    Button b4("Solve", 100, 350,160,30);
    Button b5("Quit", 100, 450,160,30);

    MathRep mr; // Mathematical representation of circuit
    CanvasContent cc(&mr); // everything on canvas except buttons

    while(true){
        int cpoint = getClick();
        if (b1.in(cpoint)) cc.addNode();
        else if(b2.in(cpoint)) cc.addConductance();
        else if(b3.in(cpoint)) cc.addSource();
        else if(b4.in(cpoint)) mr.solve();
        else if(b5.in(cpoint)) break;
    }
}
```



So, this is our program. I am not going to show you show it to you in great detail right now. But let us compile it and let us run it. Let me bring it a little bit towards the centre. All right. So, this is the main canvas created by the program. And you can see these four rectangles which are buttons they are called buttons because if I click on them then some action will happen. And each button typically has a label. So here the labels are node, conductance, source, solve and quit and then labels are there just to tell you what the function of the button is. So this button called node allows us to create nodes in this which will be parts of our circuit.

So, let us start with that. So, I am going to click on node and now I am allowed to click on the canvas and then that will create a node. So, for example this has created a node which has been numbered 0. I want more nodes. So, I will click once more. Now I get node 1. I will click one more time and I will get Node 2. So, I am going to have a very simple circuit just three nodes and three components. So, I do not need any more nodes. So now I want to add conductances.

So, I click on conductance and now I am expected to click on two nodes which have been created earlier. So, I will click on this node and I will click on this node. And as a result we will have a conductance inserted between these two nodes and furthermore you can see here that I am expected to type in the value of the conductance. So, let us say I type in 5 as the value. So you can see that that value has appeared over here. Now in the circuit diagram I showed you. There was a very specific standard symbol of resistance drawn over here.

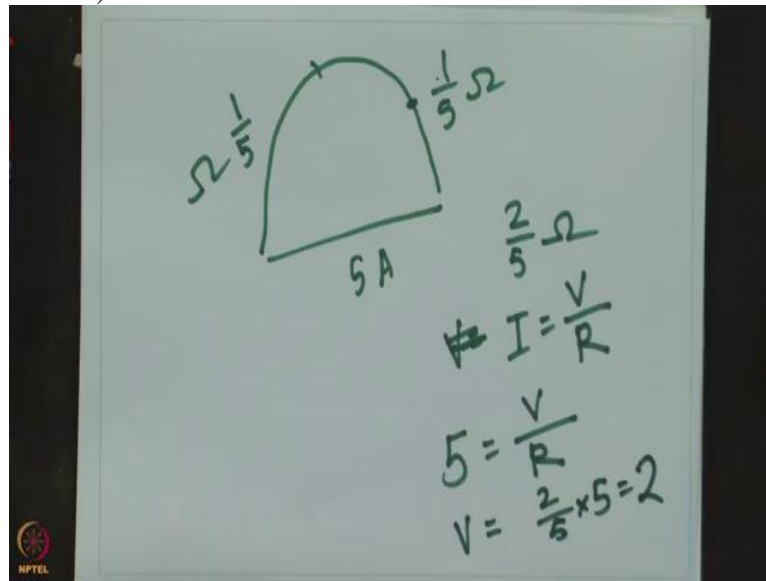
So, we can do that, we can do that. But just to keep this program simple, and because I want to illustrate the main ideas rather than the details, I have kept this quiet simple. Anyway, so what has happened is that we have been able to insert a conductance between these two nodes. I want one more conductance and let us say I put down a conductance again, so I click conductance and now I can click two nodes again. So, I click this node and I click this node, so another conductance has been entered and I have I am asked to give its value again. So, let us say I will give you the value again as 5.

So, you can see that 5 has appeared over here. Now I want to put in a source. After all, unless we have a source no current will flow. So, let us put down a source over here and let us say I am going to make the source push current from vertex 2 to vertex 1. So, I will click here first. Then I will click here. And now I am going to, I have to type in the source strength. So let us say my source strength is 5 or this is going to force 5 amperes of current through the circuit. And by the way the conductance values were mhos, or 1 over ohms, if you like.

So, what has happened now is that this current source has been placed over here. Its value is 5. And as you can see the direction has been pointed out. So, the direction is going from 2 to 1. And you may recognize this as our turtle. So, I put down the turtle over here. But I really should put down something more consistent with the voltage source, like a current source picture like the one we saw earlier. But again, I wanted to illustrate the main ideas and this was sort of the easiest way of doing things. So we will see this, how exactly this has been done but what I have done is that I have imprinted the turtle so as to give the direction of the flow of the current.

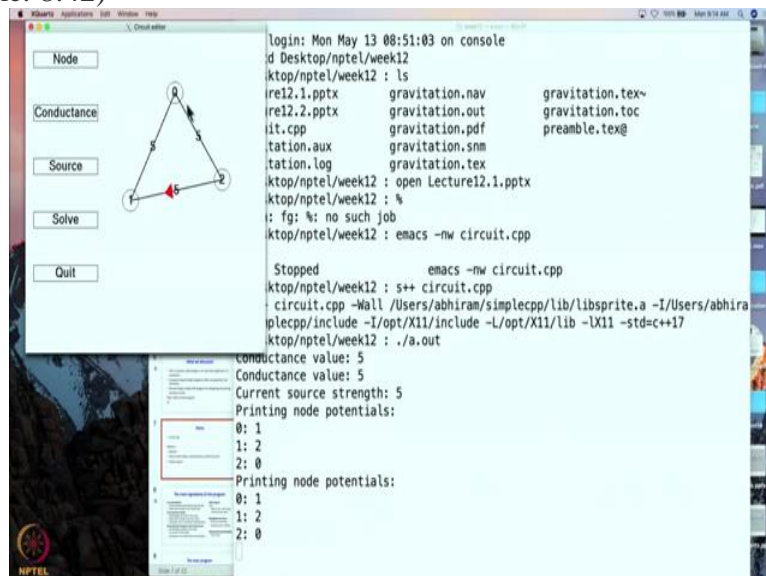
So, now say we ask this to be solved. So, if we solve this, so I have to click on this. So, if I click on this, what happens is I get the potentials at the various nodes, I guess I clicked it twice and therefore I got the potentials two times. But anyway, so what has happened? So, this says that node 0 has 1v potential, 2 has 0v. And 1 has potential 2, ok. So I think that these values are inverse but they are actually resistances. And so I am pushing, so I am pushing a current of 5 amperes through this. Sorry, they are mhos and so if I push a current through 5 amperes so let us just calculate what happens.

(Refer Slide Time: 7:42)



So I have a 5 ampere current source and it flows through 1 over 5 ohms resistance. And over here also 1 over 5 ohms resistance. So, what does this do? So, the total resistance is 2 over 5 ohms. And so the current equal to voltage upon the resistance. And so we have 5 equals the voltage upon resistance. And so, the resistance and the resistance is 2 upon 5. So, the total voltage drop is 2 upon 5 times 5 or equal to 2 volts.

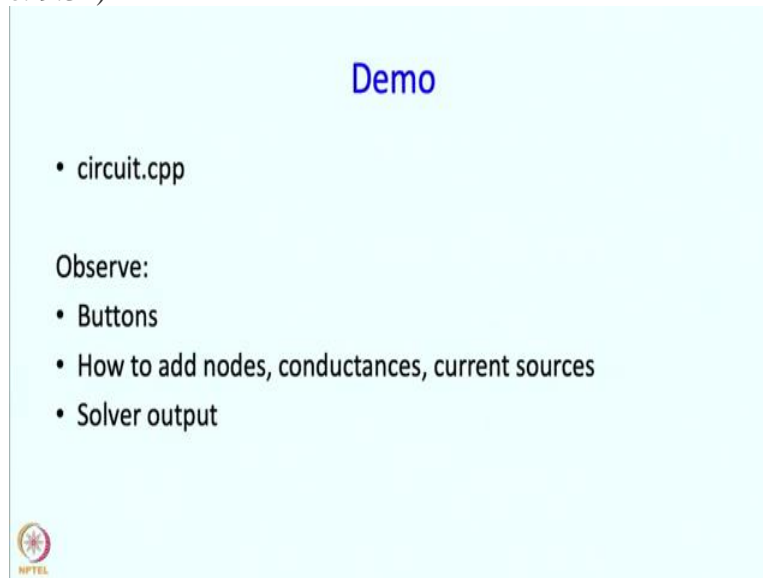
(Refer Slide Time: 8:42)



So, what has happened over here? The total drop from here to here is 2 volts as we calculated. And as you can see the potential at 1 is 2 and the potential at 2 is 0. So indeed there is a total drop of 2 volts going from here to here. And furthermore the potential in the middle is exactly half and that is to be expected because we have half the resistance over here and half the

resistance over here as well. So, our program has worked, it has solved the circuit, it has allowed us to design the circuit and solve it and it has solved it as per our expectation. So at this point we can quit the program, for that we press the quit button. So, now I can go back to our presentation.

(Refer Slide Time: 9:32)




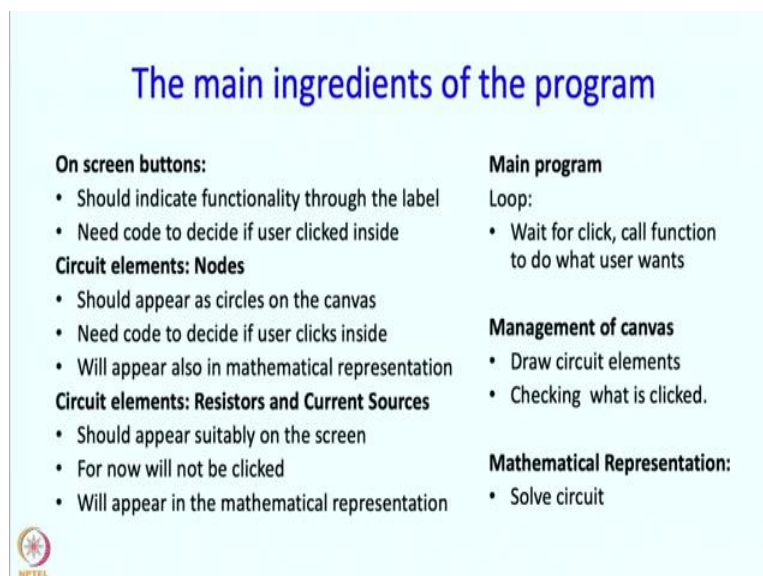
Demo

- circuit.cpp

Observe:


- Buttons
- How to add nodes, conductances, current sources
- Solver output





The main ingredients of the program

<p>On screen buttons:</p> <ul style="list-style-type: none"> • Should indicate functionality through the label • Need code to decide if user clicked inside <p>Circuit elements: Nodes</p> <ul style="list-style-type: none"> • Should appear as circles on the canvas • Need code to decide if user clicks inside • Will appear also in mathematical representation <p>Circuit elements: Resistors and Current Sources</p> <ul style="list-style-type: none"> • Should appear suitably on the screen • For now will not be clicked • Will appear in the mathematical representation 	<p>Main program</p> <p>Loop:</p> <ul style="list-style-type: none"> • Wait for click, call function to do what user wants <p>Management of canvas</p> <ul style="list-style-type: none"> • Draw circuit elements • Checking what is clicked. <p>Mathematical Representation:</p> <ul style="list-style-type: none"> • Solve circuit
---	--



So, let us continue. So, what are the main ingredients of the program? Well, the immediate ingredient to hit you is the onscreen buttons. And we observed that there is a label and that label is meant to indicate the functionality. And this means of course, that in our program we need code to decide whether the user clicked inside a button. So, if the user clicked inside the button, we need to take certain action depending upon which button the user clicked. Then there were these circuit elements, nodes. So, these appear as circles on the canvas. And we

also put down their number inside the node itself. And again, we had to click the buttons, click the nodes when we were inserting components.

So, again we need code to decide whether a click is inside a node or where is it. And nodes will also appear in the mathematical representation of the circuit. So after all, we just do not want to do a drawing. We want to solve the circuit. So, what is going to happen? Well. A drawing will get made, but as the drawing is made, we will be building up a representation of the circuit itself. So, collecting all its data and putting it into a nice convenient data structure.

So nodes will appear in that data structure as well. And exactly how they appear, I will talk about in a minute. Then there are two major circuit elements, resistors or conductances and current sources. Again they should appear suitably on the screen. And for now, in our baby program we are not going to have any occasion to click them. But in general, we may want to have, make them clickable. So, if we do that then we would need to have code to check whether clicks are lying inside these components as well. And these components will of course appear in the mathematical representation. So, let me say a little bit about how our program seems to be required to be organized based on what we just said. Well, as you saw it had a main loop. In the loop the program was waiting for the user to click. And based on what was clicked, the subsequent actions were different. So, if I clicked the node button, then clearly the program did something so that subsequently a node was created. So, presumably, it called a function and that function was the one which was responsible to enable the user to create nodes. And similarly, if you create if you clicked conductance, then a function would have to be called in order for the user to be able to create conductances. Then we needed to do some management of the canvas. So, what do I mean by that? Well, we need to draw the circuit elements. I also, I already said that we need to be able to decide whether the clicks are inside the circuit elements. So we need to decide what is clicked. And then there is the mathematical representation. So, the mathematical representation is what is needed for solving the circuit. So, this is sort of what we can say based on observing the program that we just ran.

(Refer Slide Time: 13:37)

The main program



Creates the main entities

- Buttons
- A data structure that manages the screen
- The mathematical representation of the circuit

Executes a loop

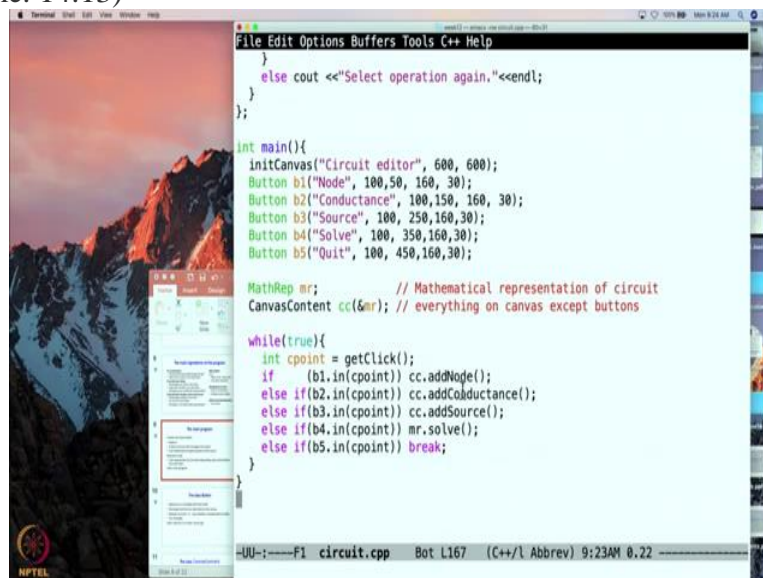
- Calls appropriate functionality depending upon what button the user clicks.

View: main program



So, let me talk about the main program. So the main program will create the main entities. So, these are the buttons, and it will create a data structure that manages the screen. And it will create a data structure that keeps track of the mathematical representation of the circuit. And indeed, it will go into a loop. So, inside the loop it will call appropriate functionality depending upon what button the user is clicking. So now we are going to take a look at the main program.

(Refer Slide Time: 14:13)




```
File Edit Options Buffers Tools C++ Help
}
else cout <<"Select operation again."<<endl;
}
};

int main(){
    InitCanvas("Circuit editor", 600, 600);
    Button b1("Node", 100,50, 160, 30);
    Button b2("Conductance", 100,150, 160, 30);
    Button b3("Source", 100, 250,160,30);
    Button b4("Solve", 100, 350,160,30);
    Button b5("Quit", 100, 450,160,30);

    MathRep mr; // Mathematical representation of circuit
    CanvasContent cc(&mr); // everything on canvas except buttons

    while(true){
        int cpoint = getClick();
        if (b1.in(cpoint)) cc.addNode();
        else if(b2.in(cpoint)) cc.addConductance();
        else if(b3.in(cpoint)) cc.addSource();
        else if(b4.in(cpoint)) mr.solve();
        else if(b5.in(cpoint)) break;
    }
}

--UU-- F1 circuit.cpp Bot L167 (C++/I Abbrev) 9:23AM 0.22
```



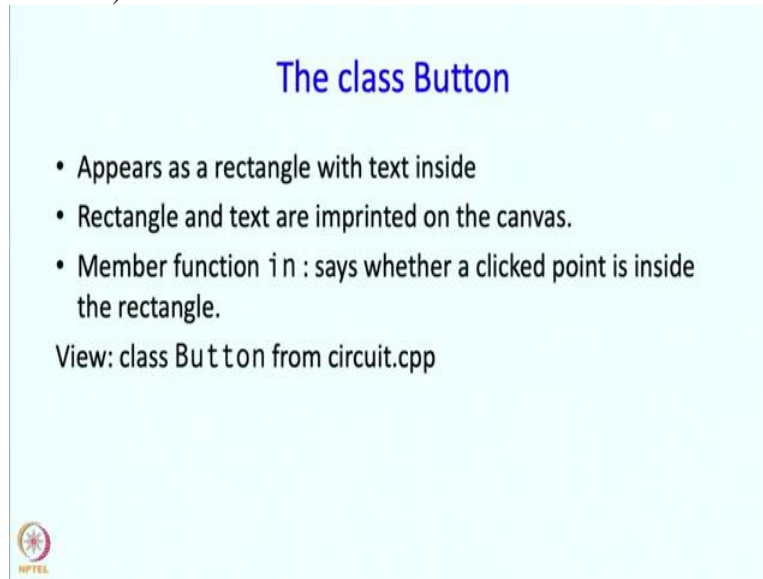
So, here is the main program. So as you saw, as we discussed it sort of creates the main things. So, what are the main things? Well, first of all, obviously it has to create the canvas. And then it has to create these buttons which you saw. And then it has to create the

mathematical representation and a data structure which manages the drawing on the canvas. As this data structure, ok. So, in the process of managing the canvas, we also have to modify the math representation. And therefore, we will pass it a pointer to the math representation as well, so that once it modifies the content on the screen it will also add relevant information to our math representation.

Then there is the main loop. So the main loop has the program waiting for a click. And at this point a click is expected only in one of these buttons. So, if you click button node then you are going to ask your canvas content data structure to get a node added. Similarly, if you click in b2 which is this button, the user is telling the program that look, I want a conductance added and therefore this function will be called. Similarly, a source addition function will be called. And if you click this b4 button then you want things solved. You want the circuit to be solved.

Well, if you want the circuit to be solved, then you may directly we may directly call the solve member function in the math representation. And finally, if button b5 is called, which is the quit button then we want to break and after breaking the entire program will terminate. So what has, so some remarks about this main program. As you can see, the main program is at a very high level. So, it is creating buttons. How exactly the buttons are created? So, R is a rectangle being drawn, some text being put inside it. That is not discussed at this level. So the constructor is called and we will see the constructor in a minute. But the constructor is going to handle the job. Yes, the constructor needs to be passed some values and those values are being passed and we will see what exactly those values mean. And then again, so the main program is creating all these entities. And again, as I said, the creation is the actual creation is in the constructor which are in the definitions of the entities. So, at this level we are staying quite high. We are staying at a high level, and we are just setting things up and the details we are not worrying about and similarly over here. Exactly what happens? What needs to happen in order to add a conductance, that is not discussed over here. That is a part of this member function. So, let us get back to the presentation.


(Refer Slide Time: 17:53)



The class Button

- Appears as a rectangle with text inside
- Rectangle and text are imprinted on the canvas.
- Member function `in` : says whether a clicked point is inside the rectangle.

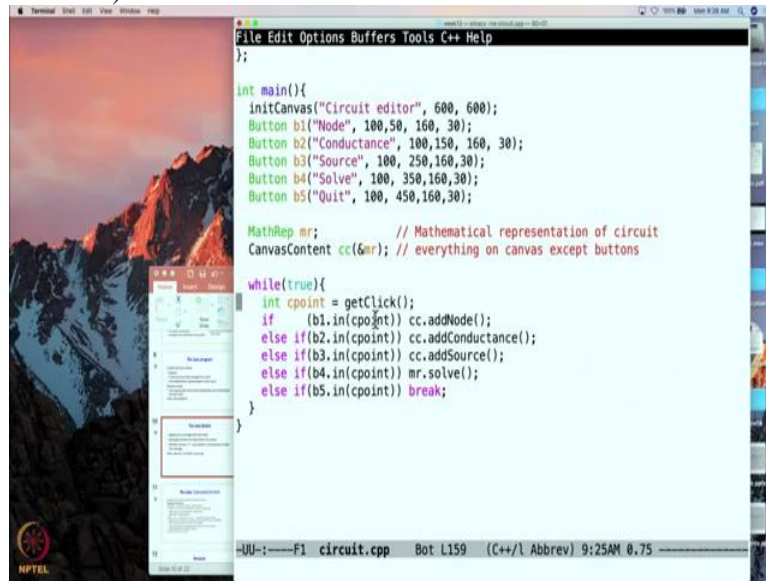
View: class `Button` from `circuit.cpp`



So, next we are going to discuss the class button. So, this has to appear as a rectangle with text inside it. And we are going to take the following approach. So the rectangle and text could be made graphic, graphics objects. So we could have, we could create graphics object of type rectangle and of type text. But in this program that is not needed because once we create that rectangle we are not going to have any occasion to move it around. Similarly, the text is not going to change. If that were changing, then it would make sense to create these as graphic objects.

So, to keep the functionality simple we are simply going to imprint. So, we are simply going to imprint the rectangle and the text on the canvas. So, we will see this in a minute and then we will have a member function in which will say whether a click point is inside the rectangle. So, that is how we will get to know whether a button has been clicked. So, let us now view the class button from our program.

(Refer Slide Time: 19:08)



```
File Edit Options Buffers Tools C++ Help
};

int main(){
    InitCanvas("Circuit editor", 600, 600);
    Button b1("Node", 100, 50, 160, 30);
    Button b2("Conductance", 100, 150, 160, 30);
    Button b3("Source", 100, 250, 160, 30);
    Button b4("Solve", 100, 350, 160, 30);
    Button b5("Quit", 100, 450, 160, 30);

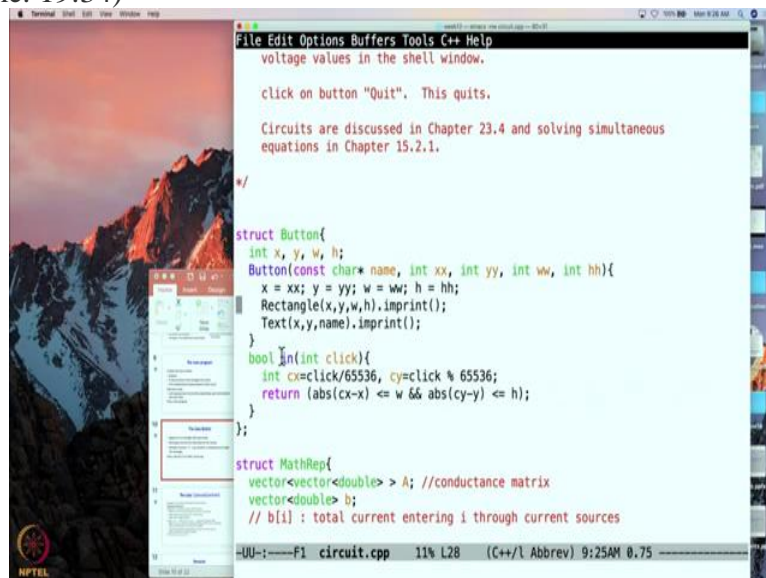
    MathRep mr; // Mathematical representation of circuit
    CanvasContent cc(&mr); // everything on canvas except buttons

    while(true){
        int cpoint = getClick();
        if (b1.in(cpoint)) cc.addNode();
        else if(b2.in(cpoint)) cc.addConductance();
        else if(b3.in(cpoint)) cc.addSource();
        else if(b4.in(cpoint)) mr.solve();
        else if(b5.in(cpoint)) break;
    }
}
```

-UU-:-----F1 circuit.cpp Bot L159 (C++/l Abbrev) 9:25AM 0.75

So, this was our main program. And as you can see the main program is creating each button and it is passing some numbers and then the button is going to be called with the member function in. So, in is going to be passed what value the click returns. So, let us see how all this is going to work.

(Refer Slide Time: 19:34)



```
File Edit Options Buffers Tools C++ Help
voltage values in the shell window.

click on button "Quit". This quits.

Circuits are discussed in Chapter 23.4 and solving simultaneous
equations in Chapter 15.2.1.

*/

struct Button{
    int x, y, w, h;
    Button(const char* name, int xx, int yy, int ww, int hh){
        x = xx; y = yy; w = ww; h = hh;
        Rectangle(x,y,w,h).imprint();
        Text(x,y,name).imprint();
    }
    bool in(int click){
        int cx=click/65536, cy=click % 65536;
        return (abs(cx-x) <= w && abs(cy-y) <= h);
    }
};

struct MathRep{
    vector<vector<double>> > A; //conductance matrix
    vector<double> b;
    // b[i] : total current entering i through current sources
}
```

-UU-:-----F1 circuit.cpp 11% L28 (C++/l Abbrev) 9:25AM 0.75

So, this is our class for representing buttons. So, it has data members x, y, w and h. And these data members are going to keep track of the centre of the button. So, that is the x y will be the centre of the button. And w will be the width, h will be the height. Now here is the constructor which is going to be called with the title that should appear and then the x, y, w and h are passed as parameters. So these parameters are stored into the data members and

then we create the rectangle. As you might remember, to create the rectangle, we need to specify the centre coordinates and the width and the height. And then we need to, and we said we are going to imprint it. So, by the way this is also a constructor call in case you have not realized it. So, I do not have to construct an object by writing rectangle-space-name-space-parameter, name and then parameters. I can write it as rectangle of the parameter list. And this will give me a constructed object right here. But it does not have a name, but I do not want the name. I just want to imprint it, so I can just dot imprint over in this matter. And similarly, the text is also going to be created and imprinted. Then the member function in is quite simple.

So, we are getting the click value returned by a click. And as you remember, we have to get the most significant 16 bits or divide by 65536 to get the x value of the click position and take the remainder modulo 65536 to get the y value of the click position. And now, if we want to know whether this click is inside, well, for the click to be inside we need it to be the case that the distance the x distance from the clicked position to the centre. The x coordinate of the centre has to be smaller than w. And the y distance between the centre and the click position should also be smaller than the height.

So if both these conditions are satisfied and only if both these conditions are satisfied, then we know that the click is inside the rectangle and therefore we check this condition and return the result. So, if this condition is true, we return true; if this condition is false, we return false which is exactly what we wanted. So, this is how the button class works. So, again the important point is that the functional details are inside the class, and not in the main program. So this way in the main program we can think at a high level. So let us go back to our presentation.

(Refer Slide Time: 22:41)

The class Button

- Appears as a rectangle with text inside
- Rectangle and text are imprinted on the canvas.
- Member function `in` : says whether a clicked point is inside the rectangle.

View: class `Button` from `circuit.cpp`



The class CanvasContent

Manages canvas drawing and input other than buttons.

Operations performed:

`addNode` : in response to click on "Node" button.

- Waits for user to click and imprints a circle at clicked point.
- Keeps track of circle position etc. in `Node` class.
- Adds node to object `mathrep`
- `addSource`, `addConductance` : in response to button clicks.
- Waits for user to select two nodes by clicking on them (`selectNode()`)
- Gets conductance/current value from keyboard.
- Imprints appropriate representation on screen. (turtle for source!)
- Adds components into object `mathrep`

View: class `CanvasContent`



So, this is what we saw. And now let us take a look at the class `CanvasContent`. So, the `CanvasContent` class manages the canvas drawing and input other than buttons. So buttons, we have buttons are special because buttons have this control like function. So, we have kept them separately but the other circuit components, how they appear on the screen and what needs to be drawn will be managed by this class.

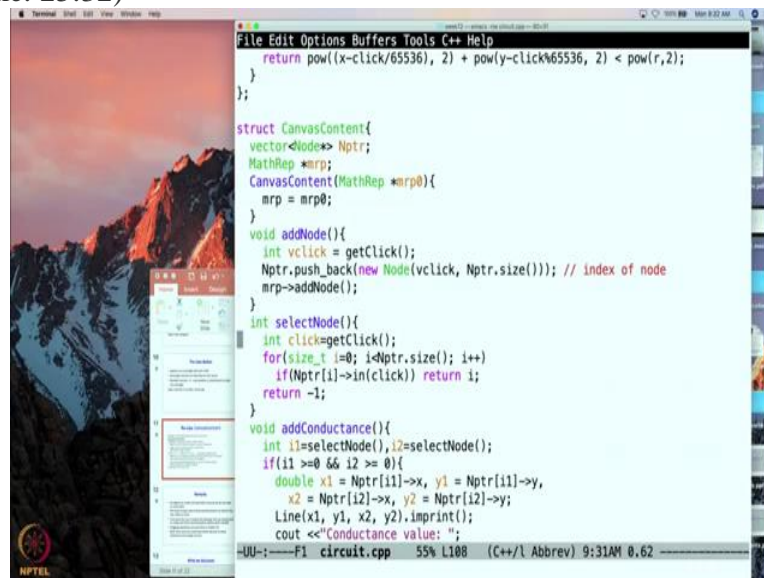
So, what operations will it perform? Well, it will have a member function `addNode` which will be called in response to click on the node button. So, we already saw that in the main program. And so the main program waits for the user to click, sorry. So this function is going

to wait for the user to click. And then after the user clicks, it is going to imprint a circle at the click point and not only just the circle but it is also going to put in the node number. So if you remember our execution of the program, this is exactly what happened.

And it is going to keep track of the circle position in this node class. Why? Because later on, we may want to know whether a click is inside a node. And we also need to add this node into the mathematical representation. So, these are the functions that our addNode member function needs to perform in response to the user saying that I want a node and clicking on the canvas to get that node. Then there are other member functions: addSource and addConductance and these also act in response to button clicks. So, here the wait, the program waits for users to select 2 nodes. And this is handled by a member function SelectNode.

And the program gets the conductance and current values from the keyboard and it imprints appropriate representation on the screen. So, once the data is obtained, once the canvas has been painted with the right representation, we have to add these components into the object MathRep as well. So, let us take a look at this CanvasContent object.

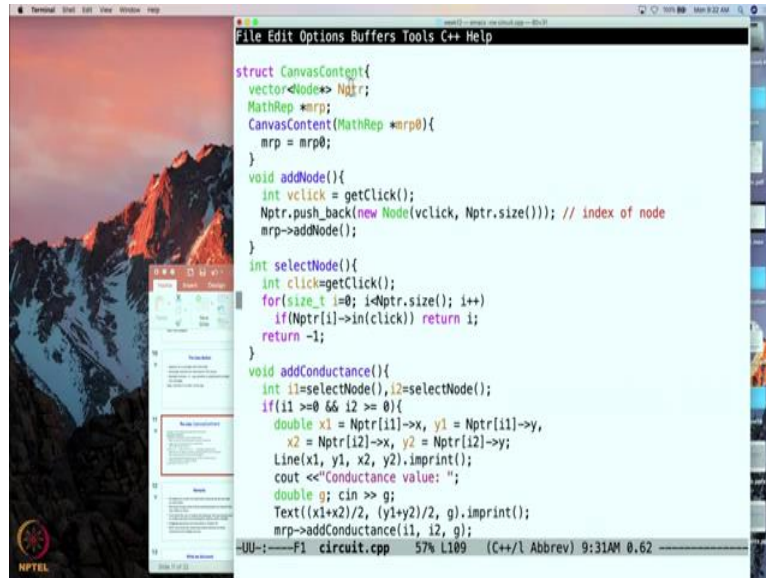
(Refer Slide Time: 25:32)



```
File Edit Options Buffers Tools C++ Help
return pow((x-click/65536), 2) + pow((y-click/65536), 2) < pow(r,2);
};

struct CanvasContent{
    vector<Node*> Nptr;
    MathRep *mrp;
    CanvasContent(MathRep *mrp0){
        mrp = mrp0;
    }
    void addNode(){
        int vclick = getClick();
        Nptr.push_back(new Node(vclick, Nptr.size())); // index of node
        mrp->addNode();
    }
    int selectNode(){
        int click=getClick();
        for(size_t i=0; i<Nptr.size(); i++)
            if(Nptr[i]->in(click)) return i;
        return -1;
    }
    void addConductance(){
        int i1=selectNode(), i2=selectNode();
        if(i1 >= 0 && i2 >= 0){
            double x1 = Nptr[i1]->x, y1 = Nptr[i1]->y,
                   x2 = Nptr[i2]->x, y2 = Nptr[i2]->y;
            Line(x1, y1, x2, y2).imprint();
            cout <<"Conductance value: ";
        }
    }
};

- UU- F1 circuit.cpp 55% L108 (C++/1 Abbrev) 9:31AM 0.62
```

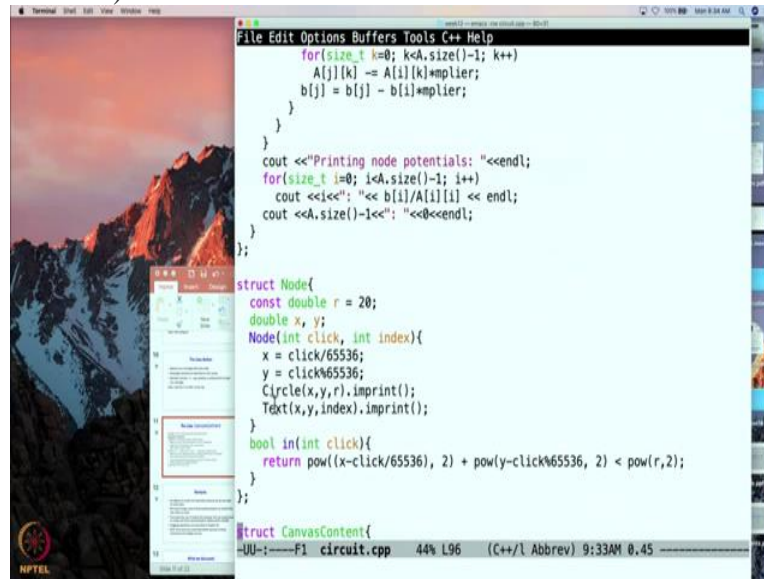

A screenshot of a C++ code editor window. The editor has a menu bar with 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'C++', and 'Help'. The code defines a class named 'CanvasContent'. It includes a vector of pointers to 'Node' objects, a pointer to a 'MathRep' object, and a constructor that takes a 'MathRep' pointer. The class has three methods: 'addNode()' which gets a click position and adds a new node to the vector and the math representation; 'selectNode()' which iterates through the nodes to find one based on a click; and 'addConductance()' which selects two nodes and adds a conductance between them. The status bar at the bottom shows 'F1 circuit.cpp', '57% L109', and '(C++/1 Abbrev) 9:31AM 0.62'.

```
struct CanvasContent{
    vector<Node*> Nptr;
    MathRep *mrp;
    CanvasContent(MathRep *mrp0){
        mrp = mrp0;
    }
    void addNode(){
        int vclick = getClick();
        Nptr.push_back(new Node(vclick, Nptr.size())); // index of node
        mrp->addNode();
    }
    int selectNode(){
        int cclick=getClick();
        for(size_t i=0; i<Nptr.size(); i++){
            if(Nptr[i]->in(cclick)) return i;
        }
        return -1;
    }
    void addConductance(){
        int i1=selectNode(), i2=selectNode();
        if(i1 >= 0 && i2 >= 0){
            double x1 = Nptr[i1]->x, y1 = Nptr[i1]->y,
                x2 = Nptr[i2]->x, y2 = Nptr[i2]->y;
            Line(x1, y1, x2, y2).imprint();
            cout <<"Conductance value: ";
            double g; cin >> g;
            Text((x1+x2)/2, (y1+y2)/2, g).imprint();
            mrp->addConductance(i1, i2, g);
        }
    }
};
```

So here is our CanvasContent object. So, what does it keep track of? Well, it has to keep track of what nodes are present on the screen. So therefore, it needs this pointer to nodes. And it also needs to know about the math representation. So it is going to have a member mrp which is going to be a pointer to the math representation. Here is the constructor class, the constructor, sorry here is the constructor. The constructor is just going to be passed with a pointer to the math representation and that pointer is going to be stored over here.

So, let us take a look at addNode. So addNode is going to ask the user to click. And once the user clicks, we are going to create a new node at that click position. So, we will see the node class in a second, but the constructor of the node will expect a click value to be passed. And, it has to decide what its id is. So, we will pass Nptr.size because currently we have had nodes going from 0 to Nptr.size-1. So, Nptr stores pointers to all the nodes created. So, the next node will indeed be created at this position. So, at this index using this index and so that is what we passed, and then we add this created node, sorry, we add a node to our math representation as well. So let us take a quick look at the node class.

(Refer Slide Time: 27:26)



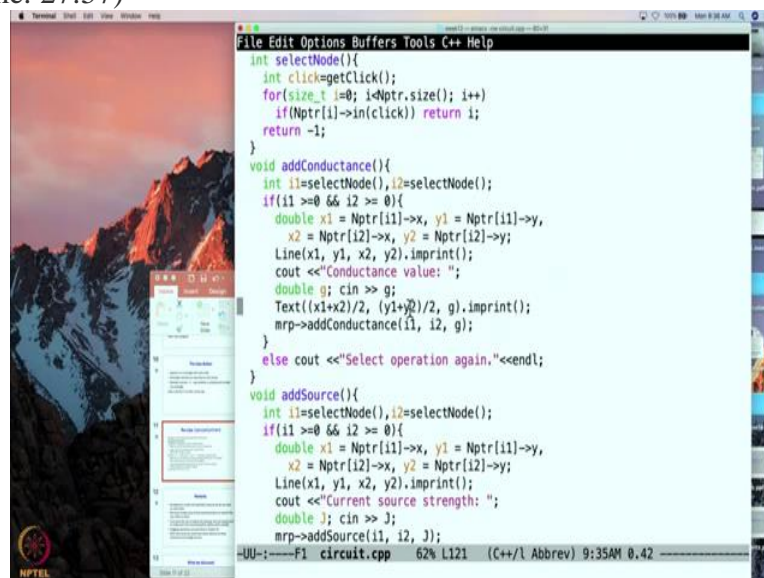
```
File Edit Options Buffers Tools C++ Help
for(size_t k=0; k<A.size()-1; k++)
    A[j][k] -= A[i][k]*mplier;
    b[j] = b[j] - b[i]*mplier;
}
}
cout <<"Printing node potentials: "<<endl;
for(size_t i=0; i<A.size()-1; i++)
    cout <<"i: " << b[i]/A[i][i] << endl;
    cout <<A.size()-1<<": " <<0<<endl;
};

struct Node{
    const double r = 20;
    double x, y;
    Node(int click, int index){
        x = click/65536;
        y = click/65536;
        Circle(x,y,r).imprint();
        Text(x,y,index).imprint();
    }
    bool in(int click){
        return pow((x-click/65536), 2) + pow((y-click/65536), 2) < pow(r,2);
    }
};

struct CanvasContent{
    -UU-:-----F1 circuit.cpp 44% L96 (C++/l Abbrev) 9:33AM 0.45
```

So, the node class is over here. It receives at the constructor, for the constructor it receives a click value and the index. It calculates where the click actually happened and then it creates a circle and imprints at that position. So r is some 20 that we fixed and it imprints the text. So, the text is just the index. So if you remember, inside the node we had the node number printed. So, this is what does it.

(Refer Slide Time: 27:57)



```
File Edit Options Buffers Tools C++ Help
int selectNode(){
    int click=getClick();
    for(size_t i=0; i<Nptr.size(); i++)
        if(Nptr[i]->in(click)) return i;
    return -1;
}

void addConductance(){
    int i1=selectNode(), i2=selectNode();
    if(i1 >=0 && i2 >= 0){
        double x1 = Nptr[i1]->x, y1 = Nptr[i1]->y,
                x2 = Nptr[i2]->x, y2 = Nptr[i2]->y;
        Line(x1, y1, x2, y2).imprint();
        cout <<"Conductance value: ";
        double g; cin >> g;
        Text((x1+x2)/2, (y1+y2)/2, g).imprint();
        mrp->addConductance(i1, i2, g);
    }
    else cout <<"Select operation again."<<endl;
}

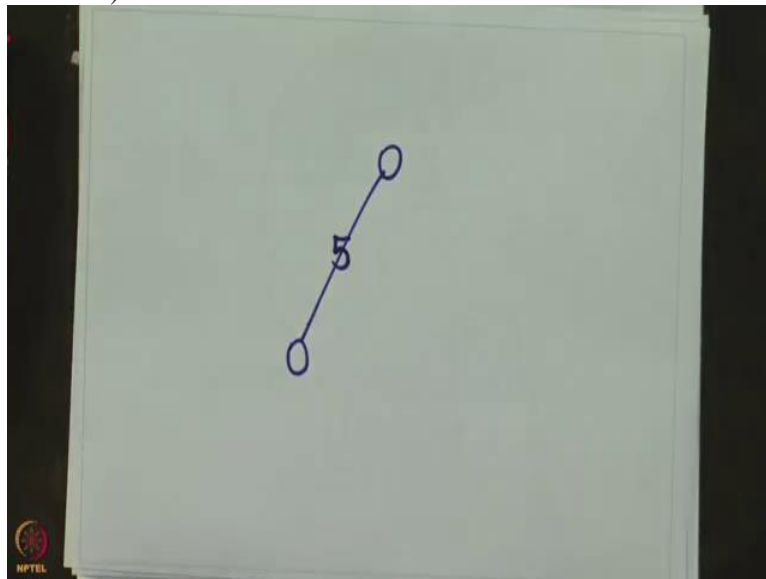
void addSource(){
    int i1=selectNode(), i2=selectNode();
    if(i1 >=0 && i2 >= 0){
        double x1 = Nptr[i1]->x, y1 = Nptr[i1]->y,
                x2 = Nptr[i2]->x, y2 = Nptr[i2]->y;
        Line(x1, y1, x2, y2).imprint();
        cout <<"Current source strength: ";
        double j; cin >> j;
        mrp->addSource(i1, i2, j);
    }
}
-UU-:-----F1 circuit.cpp 62% L121 (C++/l Abbrev) 9:35AM 0.42
```

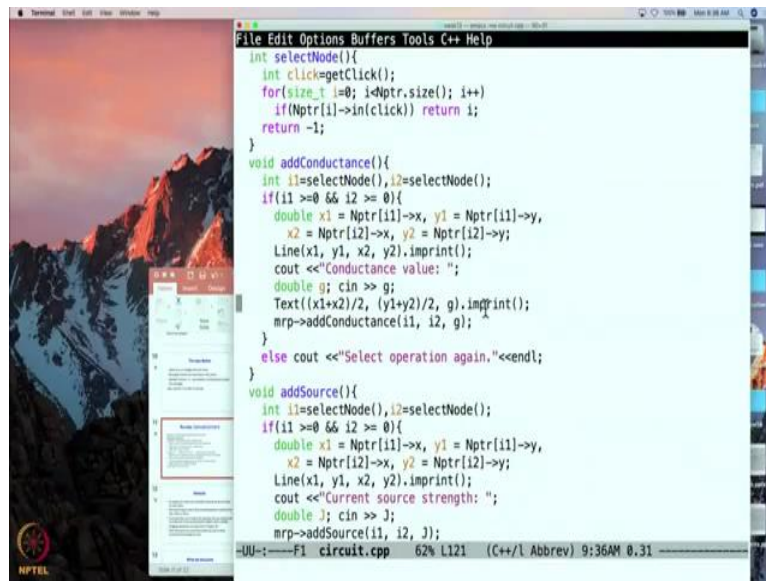
Then the select node, so let us first look at addConductance. So, addConductance is another member function inside our CanvasContent. And this is going to be called when the user clicks the conductance button. And this is the function which is responsible in helping the user to add a conductance, how does it do it? It calls the select node function twice.

So, let us see now what the select node function does. So here is the select node function. So, each call of select node waits for the user to click. And then it checks whether the click is inside a node. If it is inside a node, it will return the node index. So $i[1]$ is expected to be the index of the click node, $i[2]$ is expected to be the index of the second click node. And this is just testing whether the click was inside a node actually. If it is inside the node, now we are going to actually do the processing. That means the user has selected the two nodes between which the conductance has to be added. So, what do we do here?

So, here we are going to have to draw the line connecting those two nodes. So we get the x coordinate and the y coordinate of the first click position, x coordinate and y coordinate of the second click position and then we actually draw this line. And then on our shell window in our shell window we type out the message, give the conductance value. And so then the conductance value is received. So, then we can print, we can imprint the text at the centre of that line of that edge with the value G.

(Refer Slide Time: 30:01)





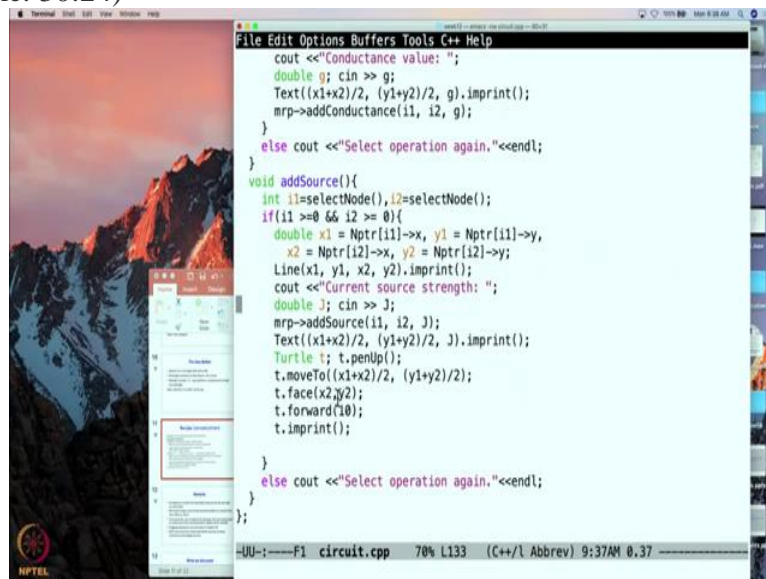
```

File Edit Options Buffers Tools C++ Help
int selectNode(){
    int click=getClick();
    for(size_t i=0; i<Nptr.size(); i++){
        if(Nptr[i]->in(click)) return i;
    }
    return -1;
}
void addConductance(){
    int i1=selectNode(), i2=selectNode();
    if(i1 >= 0 && i2 >= 0){
        double x1 = Nptr[i1]->x, y1 = Nptr[i1]->y,
               x2 = Nptr[i2]->x, y2 = Nptr[i2]->y;
        Line(x1, y1, x2, y2).imprint();
        cout <<"Conductance value: ";
        double g; cin >> g;
        Text((x1+x2)/2, (y1+y2)/2, g).imprint();
        mrp->addConductance(i1, i2, g);
    }
    else cout <<"Select operation again."<<endl;
}
void addSource(){
    int i1=selectNode(), i2=selectNode();
    if(i1 >= 0 && i2 >= 0){
        double x1 = Nptr[i1]->x, y1 = Nptr[i1]->y,
               x2 = Nptr[i2]->x, y2 = Nptr[i2]->y;
        Line(x1, y1, x2, y2).imprint();
        cout <<"Current source strength: ";
        double J; cin >> J;
        mrp->addSource(i1, i2, J);
    }
}
-UU-:-----F1 circuit.cpp 62% L121 (C++/I Abbrev) 9:36AM 0.31

```

So, as you if you might remember if the user added a conductance between say this vertex and this vertex, then the conductance value was imprinted over here. So, this is what is done in this text statement. And then we want to add this conductance into our math representation as well and that is what is being called over here. The source addition is very similar.

(Refer Slide Time: 30:24)



```

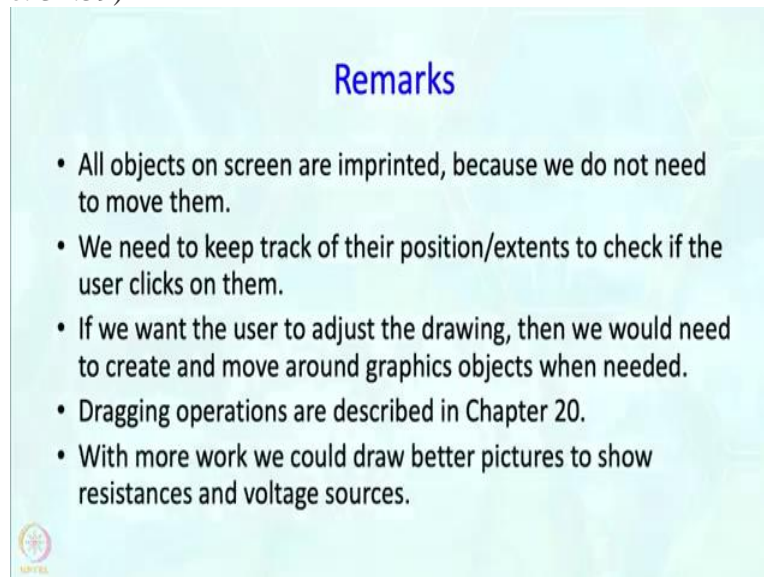
File Edit Options Buffers Tools C++ Help
    cout <<"Conductance value: ";
    double g; cin >> g;
    Text((x1+x2)/2, (y1+y2)/2, g).imprint();
    mrp->addConductance(i1, i2, g);
}
else cout <<"Select operation again."<<endl;
}
void addSource(){
    int i1=selectNode(), i2=selectNode();
    if(i1 >= 0 && i2 >= 0){
        double x1 = Nptr[i1]->x, y1 = Nptr[i1]->y,
               x2 = Nptr[i2]->x, y2 = Nptr[i2]->y;
        Line(x1, y1, x2, y2).imprint();
        cout <<"Current source strength: ";
        double J; cin >> J;
        mrp->addSource(i1, i2, J);
        Text((x1+x2)/2, (y1+y2)/2, J).imprint();
        Turtle t; t.penUp();
        t.moveTo((x1+x2)/2, (y1+y2)/2);
        t.face(x2,y2);
        t.forward(10);
        t.imprint();
    }
    else cout <<"Select operation again."<<endl;
}
}
-UU-:-----F1 circuit.cpp 70% L133 (C++/I Abbrev) 9:37AM 0.37

```

So let me just quickly observe that again we are going to select node and then we are going to draw the line, we are going to get the Source Strength and we are going to add the source into our math representation. We are going to put down the magnitude of the source and then we put the turtle. So, the turtle is going to be created, when a turtle is created it is always created at the centre of the screen. We wanted to come to the centre of that line, the line that we just drawn. So this is the centre of that line that we just drew, and here is a command. Here is I guess some undocumented command on the turtle.

And this command says make the turtle face the second end point. And then we make it move forward. So if so what this is going to do is that if this is the conductance and if this is the conductance value, the turtle will first come to this point, the centre and then it will move in the direction of the second end point and it will come over here and it will get imprinted. So, the purpose of all of this is to imprint but also imprint a little bit away from the source strength value. That is it, so that is what happens in the conductance our conductance code.

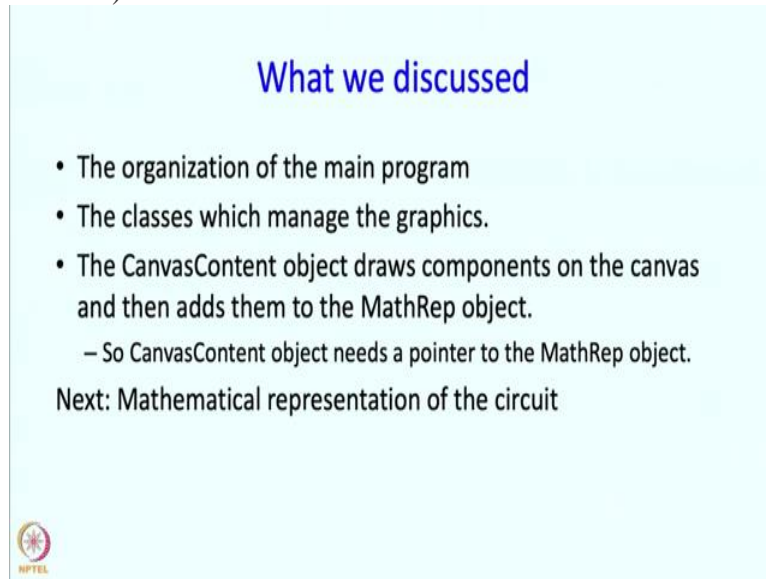
(Refer Slide Time: 31:59)



So, let us get back, so we just saw this and a few remarks: all objects on the screen are imprinted because we do not need to move them and but we do need to keep track of their positions and extents to check if the user clicks on them. If we wanted the user to adjust a drawing, then we would need to create and move around graphics objects. And if you want to dragging, which is really the nicer way of doing things, you can do it in simplecpp and that is described in chapter 20.

So, it will need a little bit more work but we really could do very nice pictures. We could draw a circuit diagram exactly like the one that I showed you at the very beginning. But it will need more work.


(Refer Slide Time: 32:44)



What we discussed

- The organization of the main program
- The classes which manage the graphics.
- The CanvasContent object draws components on the canvas and then adds them to the MathRep object.
 - So CanvasContent object needs a pointer to the MathRep object.

Next: Mathematical representation of the circuit



All right, so what have we discussed in this segment? We said, we discussed the organization of the main program, we discussed the CanvasContent class which manages the graphics and it draws components on the canvas and adds them to the MathRep. And next, we are going to talk about how we represent the circuit mathematically, and how the class MathRep is going to be designed but before that we will take a quick break.