

# Understanding Code

David A. Mellis

*Advisors*

Neil Churcher

Yaniv Steiner

Interaction Design Institute Ivrea – 21 April 2005 – Exam I

# Overview

Redesign of a debugger: software tool for **observing, annotating, and analyzing** the behavior of software.

# Agenda

*context*

*problems today*

*visual metaphors*

*design goals*

*scenario from last review*

*interface specification*

*prototype for testing*

*impact*

*next steps*

# Agenda

*context*

*problems today*

*visual metaphors*

*design goals*

*scenario from last review*

*interface specification*

*prototype for testing*

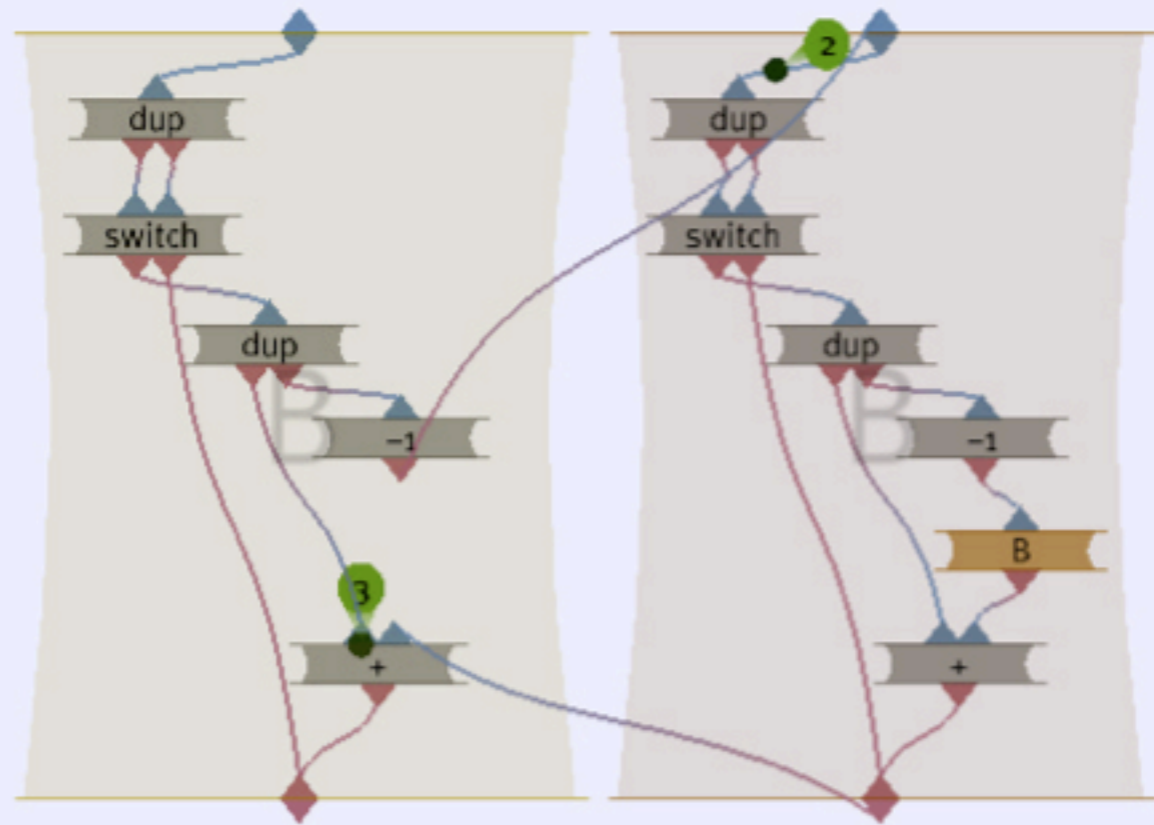
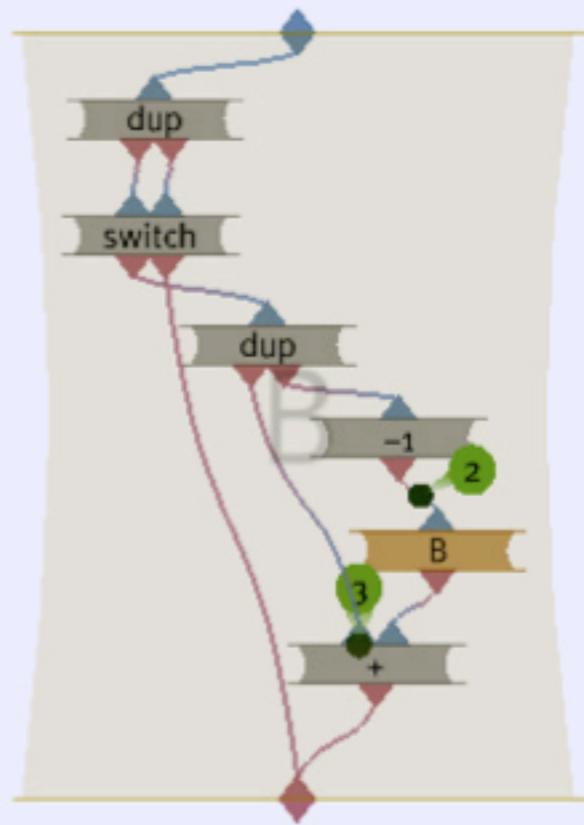
*impact*

*next steps*

## Context

What we consider a simple program today is much more complex to build. It sucks up programmer's mental processing power. They need better tools.

# What it's not: a tool for non-programmers



Pablo, a visual programming environment (MIT Media Lab).

# Agenda

*context*

*problems today*

*visual metaphors*

*design goals*

*scenario from last review*

*interface specification*

*prototype for testing*

*impact*

*next steps*

## Frustrations today

I can't keep it all in my head.

I can't tell what connects to what.

I'm getting lost in the details.

I can't see what's happening.

I don't know where in the code to look.

I don't know what will happen if I change this.

*Gathered from extended interviews with 2 professional programmers (mid-20's) and informal conversation with others.*

# Agenda

*context*

*problems today*

*visual metaphors*

*design goals*

*scenario from last review*

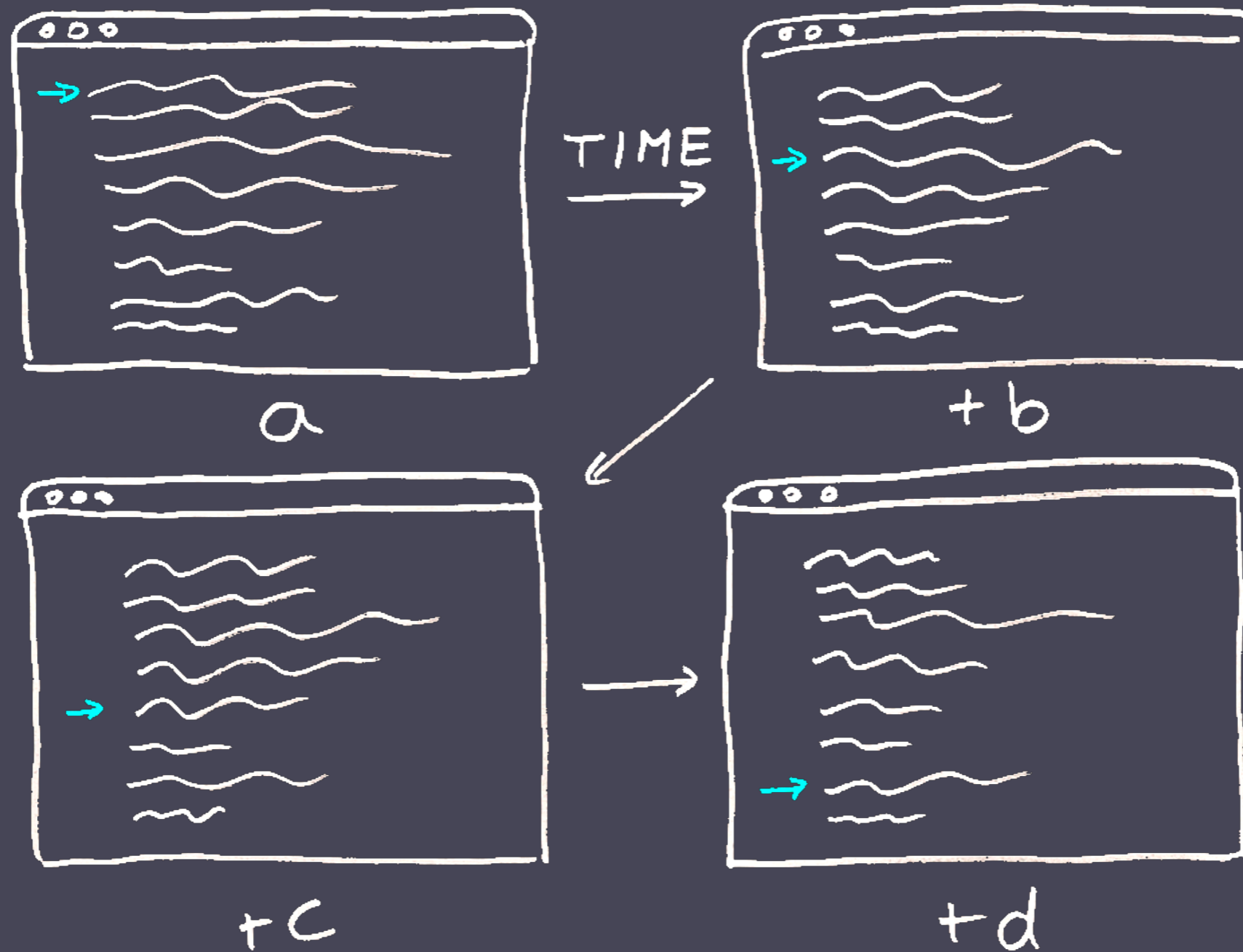
*interface specification*

*prototype for testing*

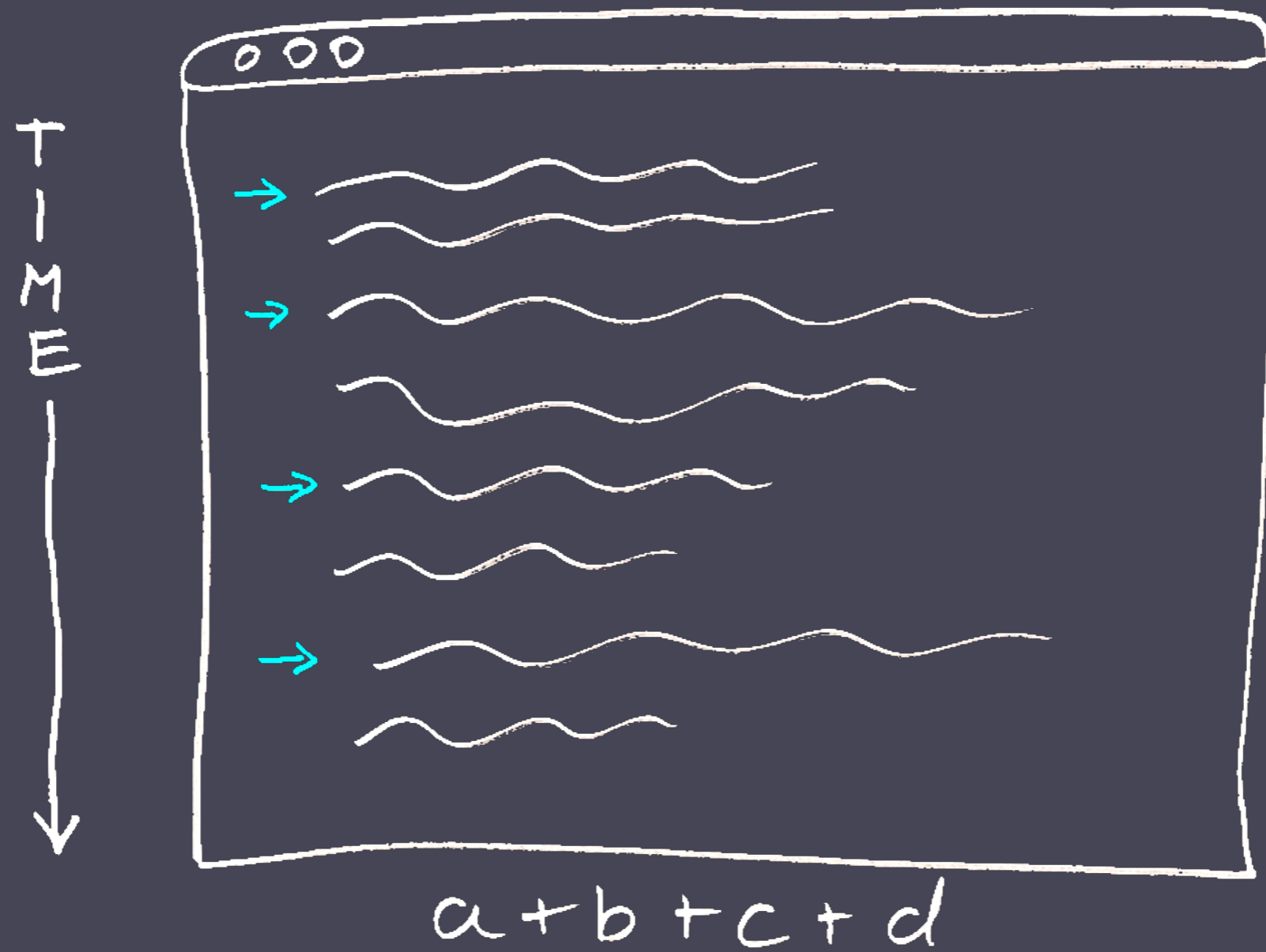
*impact*

*next steps*

# Today: freezing time in snapshots



Need: continuous timeline



Today: abstract instructions, separated data

$$a + b + c + d = ?$$

$$a = 5 \quad b = 10 \quad c = -2 \quad d = 3$$

Need: concrete operation

$$\boxed{5} + \boxed{10} + \boxed{-2} + \boxed{3} = ?$$

$a \qquad b \qquad c \qquad d$

Today: only see static structure

## GRAPHICS



Images.java



Textures.java



Lighting.java



Camera.java

## PHYSICS



Frust.java

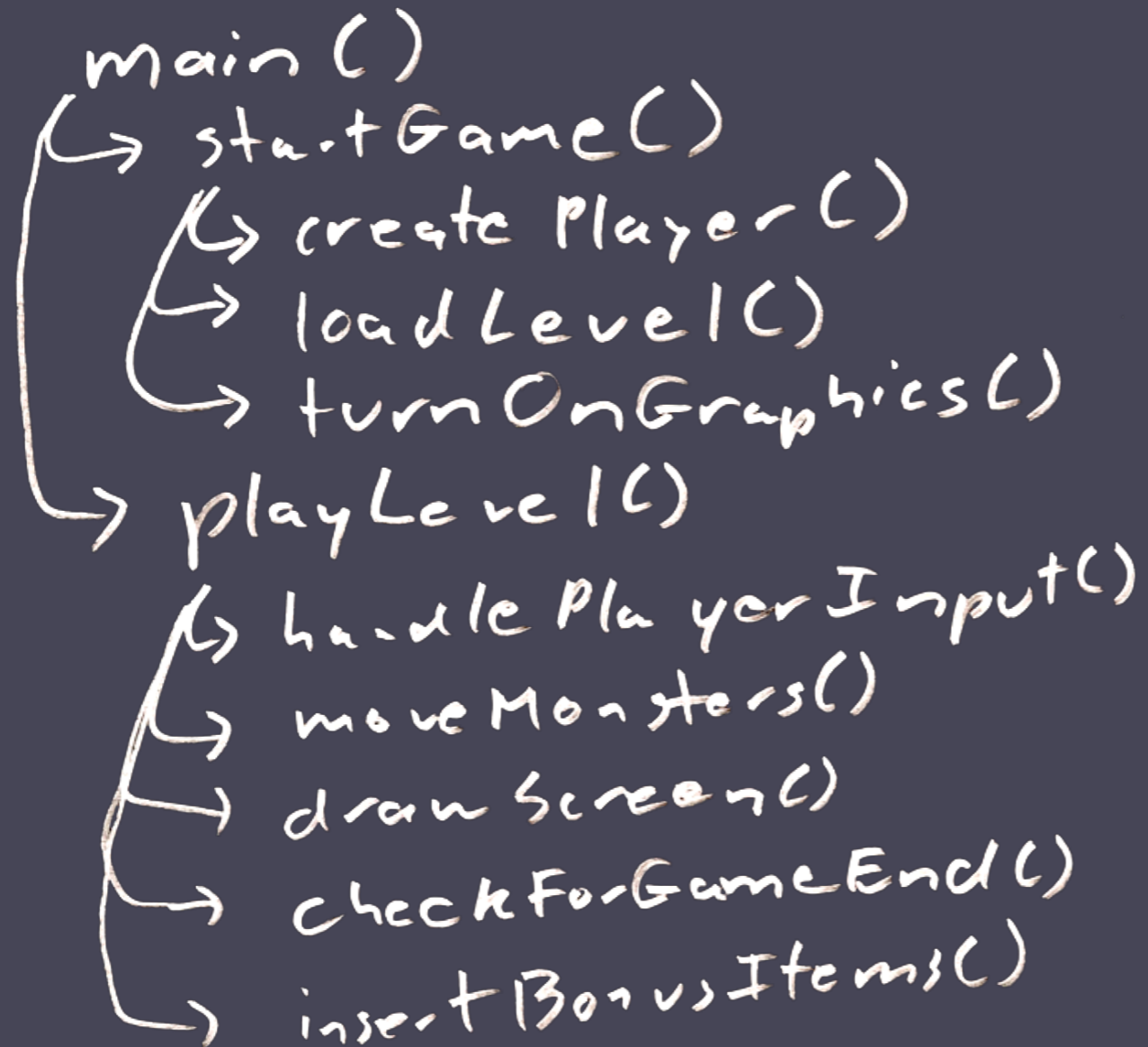


Collision.java

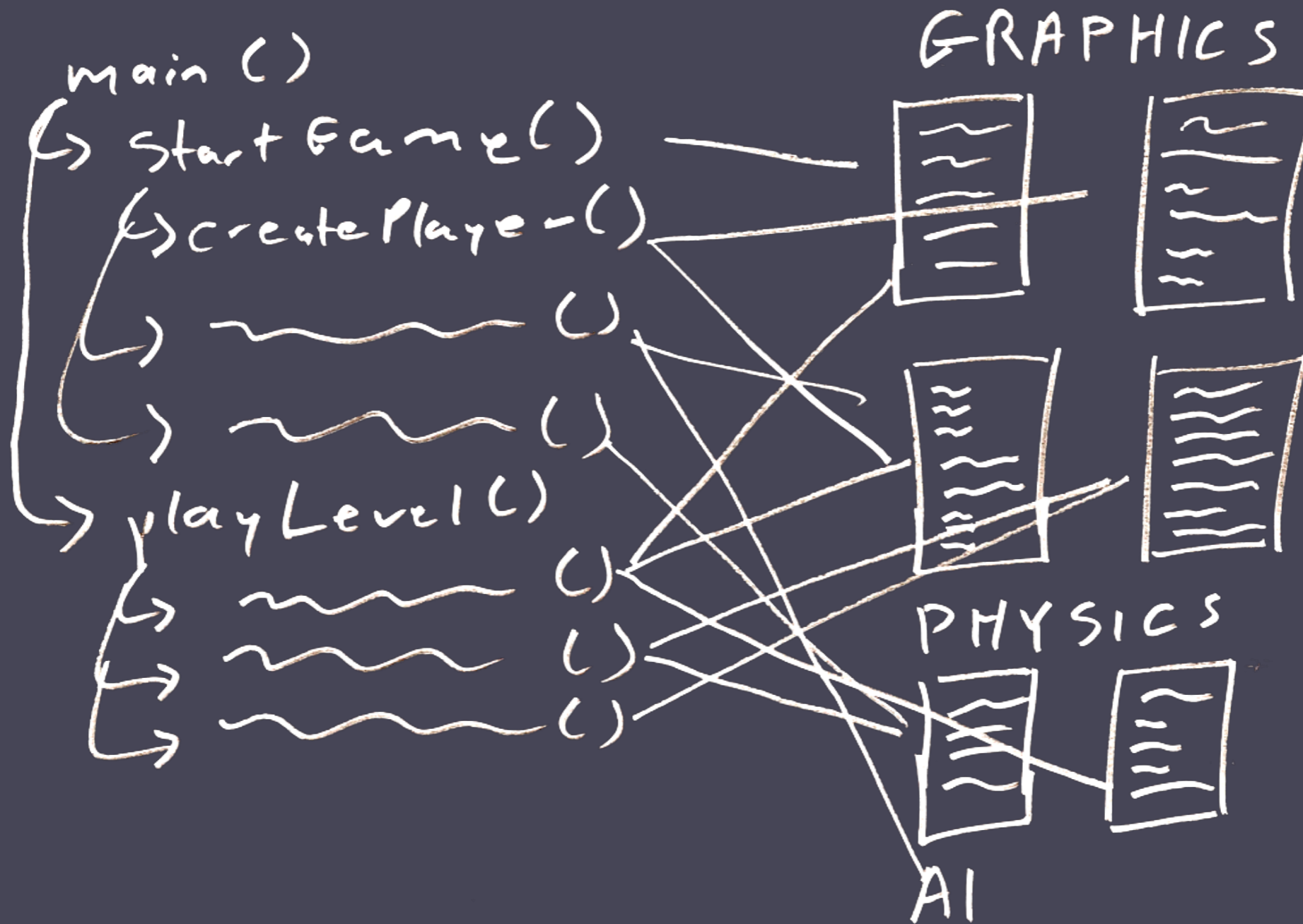
AI

GAMEPLAY

## Need: dynamic structure



Need: connections



# Agenda

*context*

*problems today*

*visual metaphors*

*design goals*

*scenario from last review*

*interface specification*

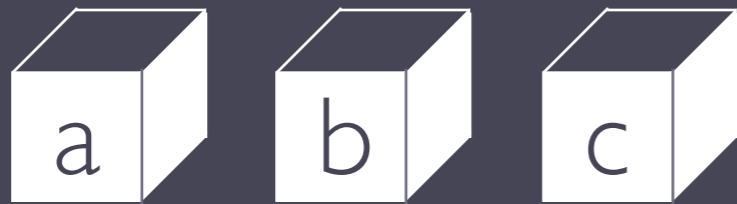
*prototype for testing*

*impact*

*next steps*

# Knowledge in the world vs. in your head

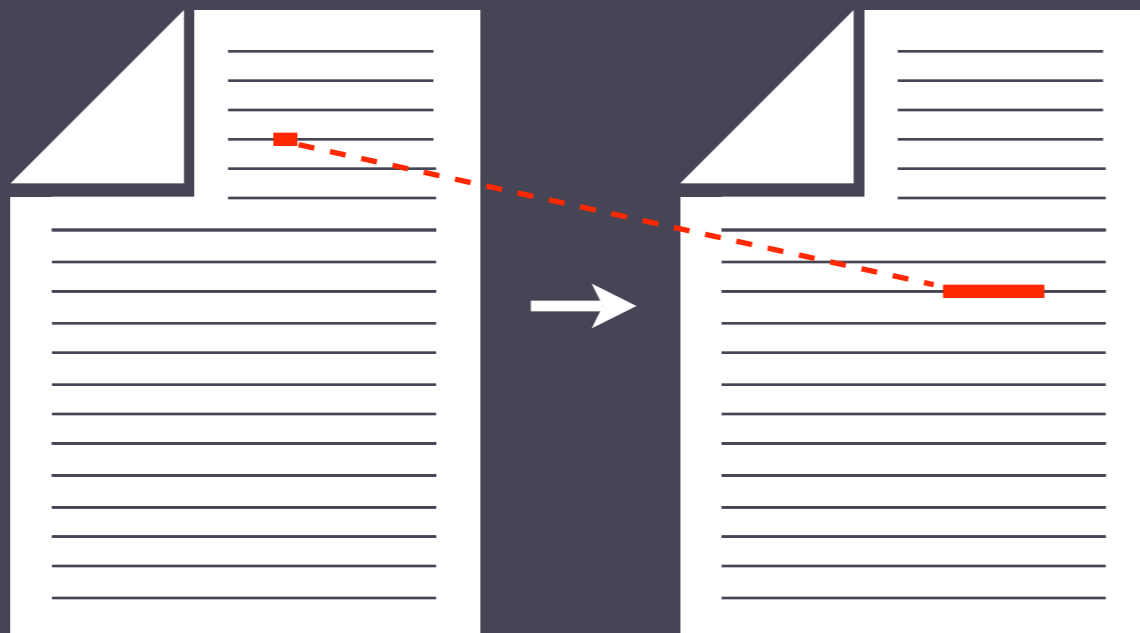
Tangible variables



Tangible time



# Tangible connections



# Supporting tasks

Debugging: focused goal, observing details



Maintenance: overall understanding, which parts do what



Complexity: details and flows across components



# Agenda

*context*

*problems today*

*visual metaphors*

*design goals*

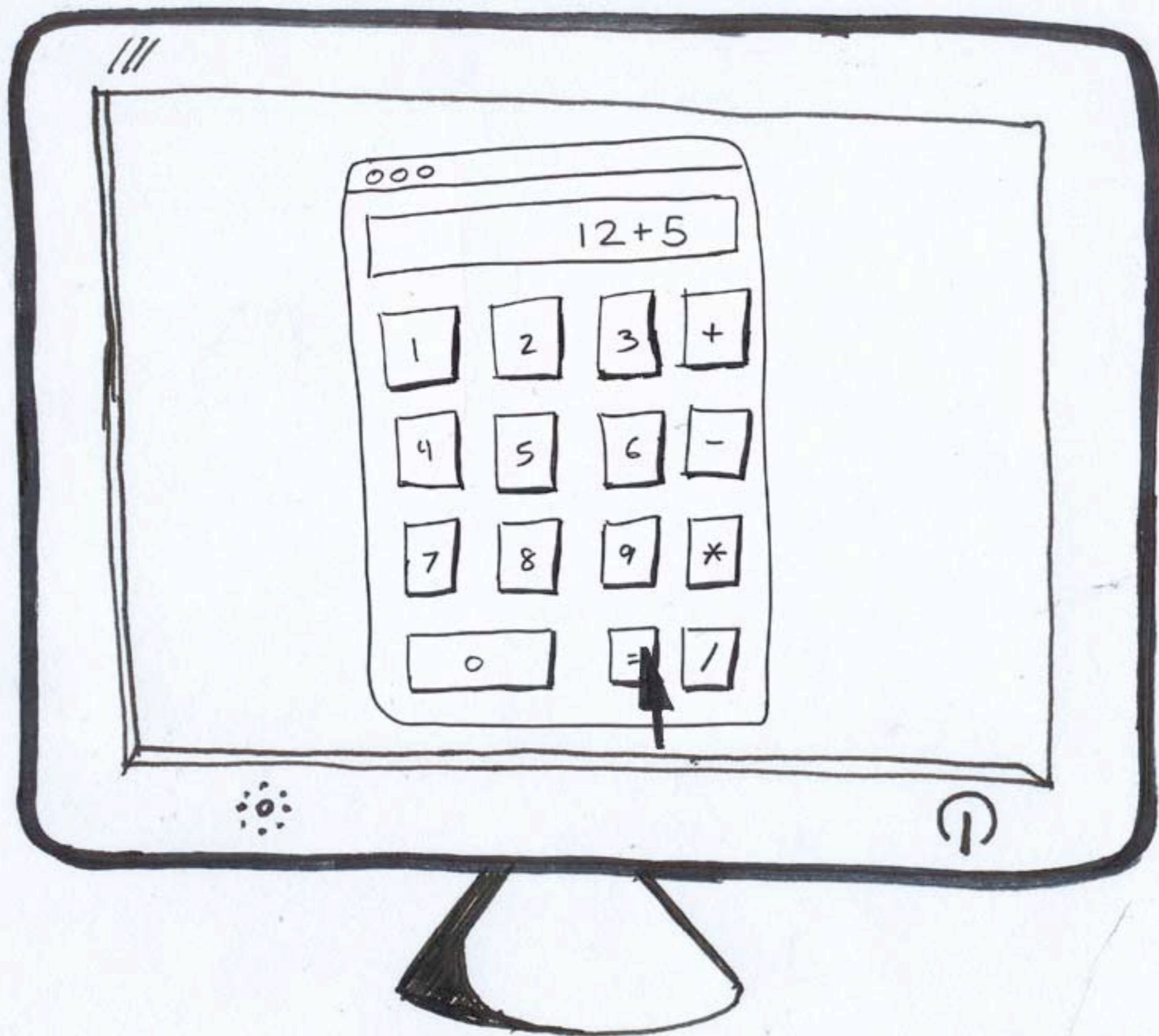
*scenario from last review*

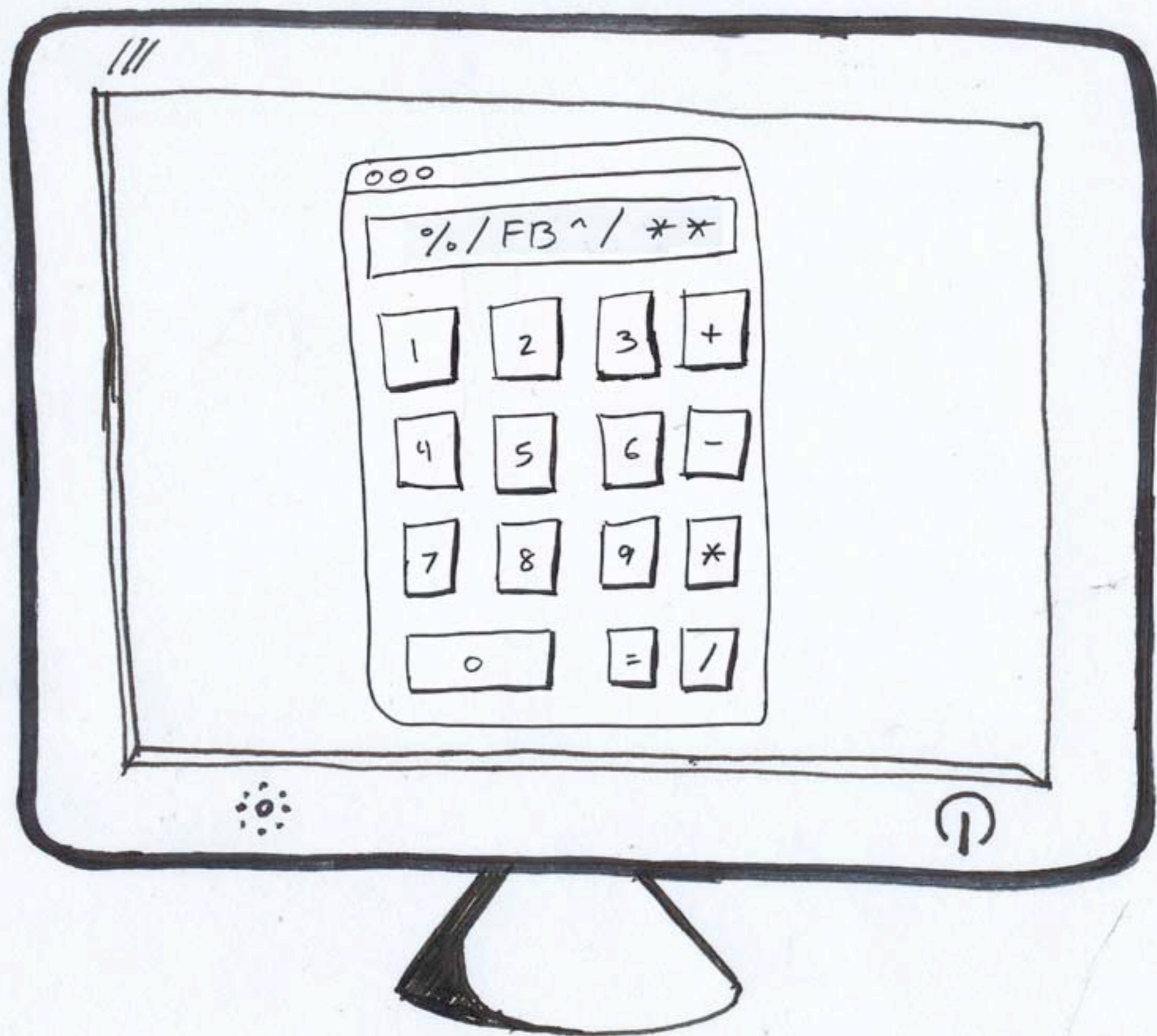
*interface specification*

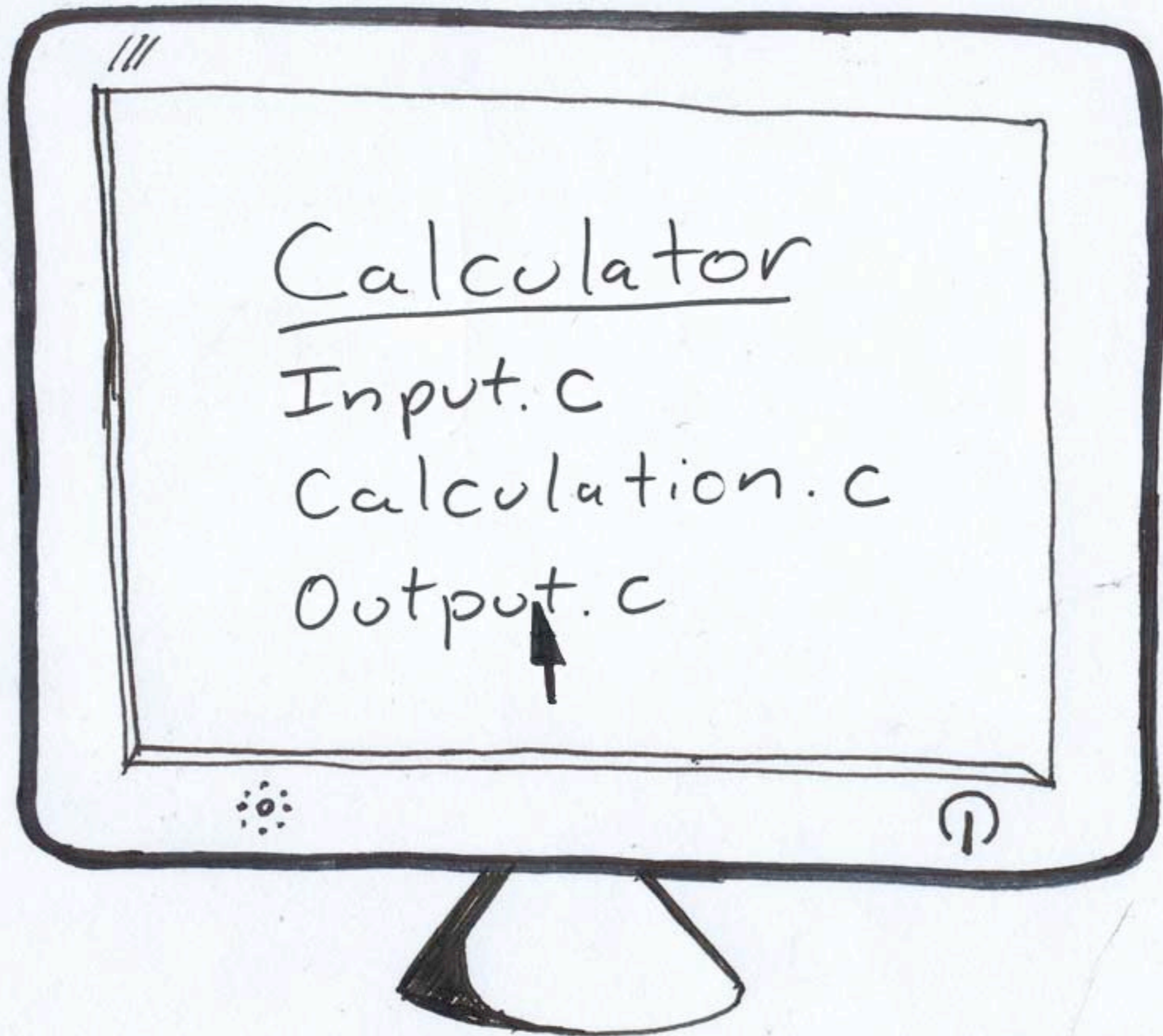
*prototype for testing*

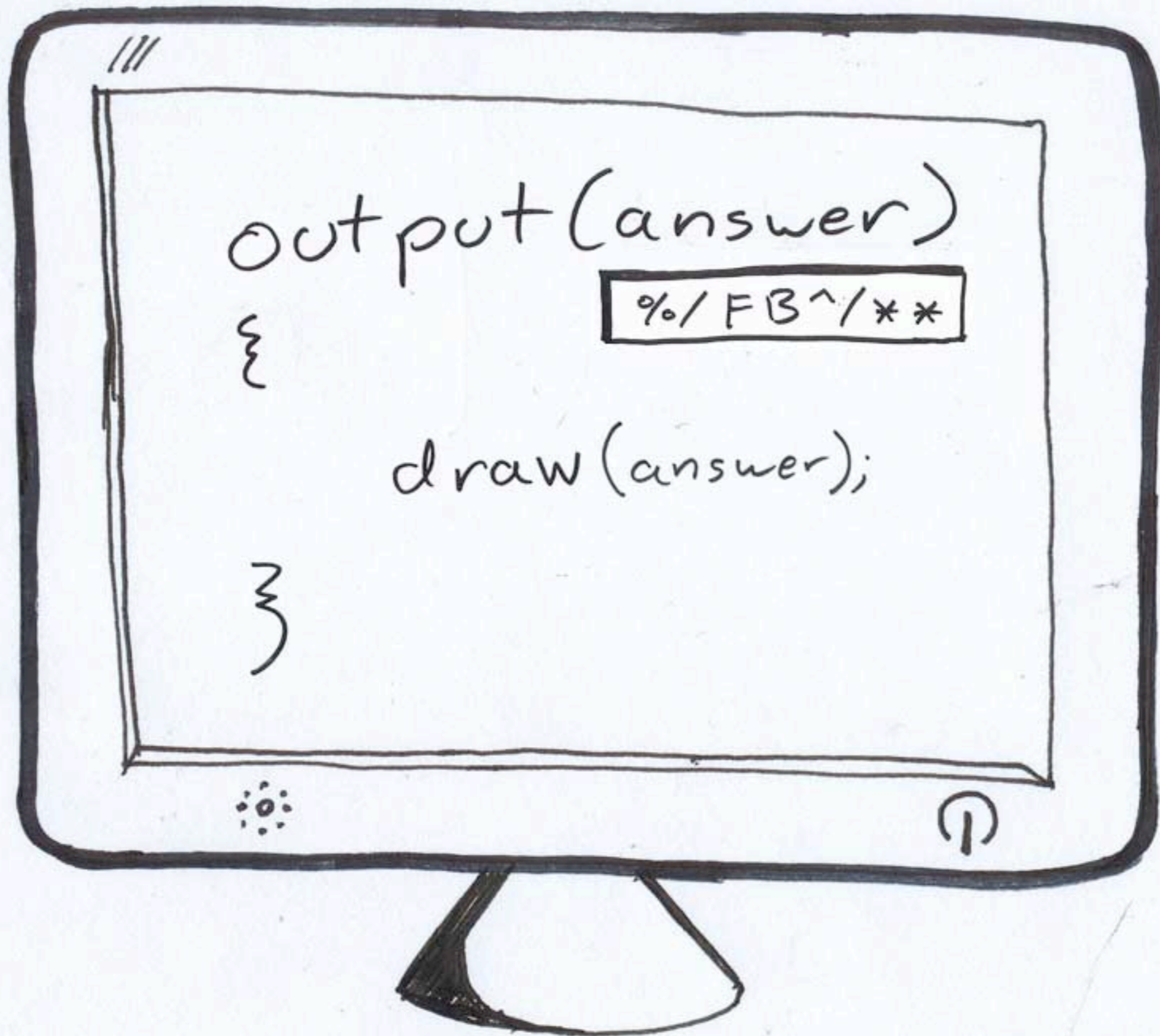
*impact*

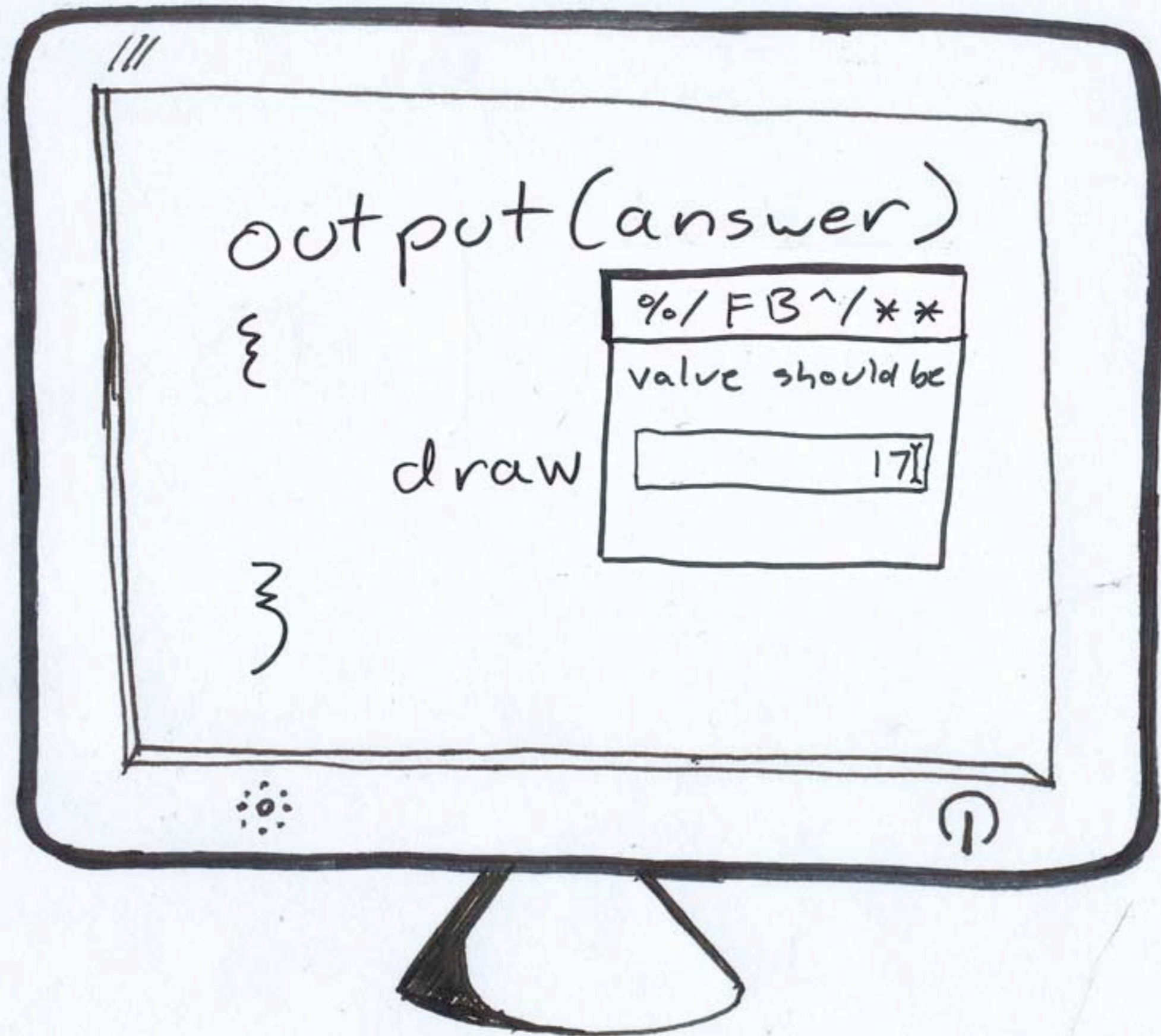
*next steps*

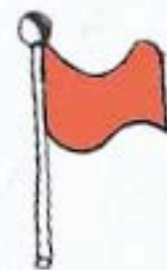
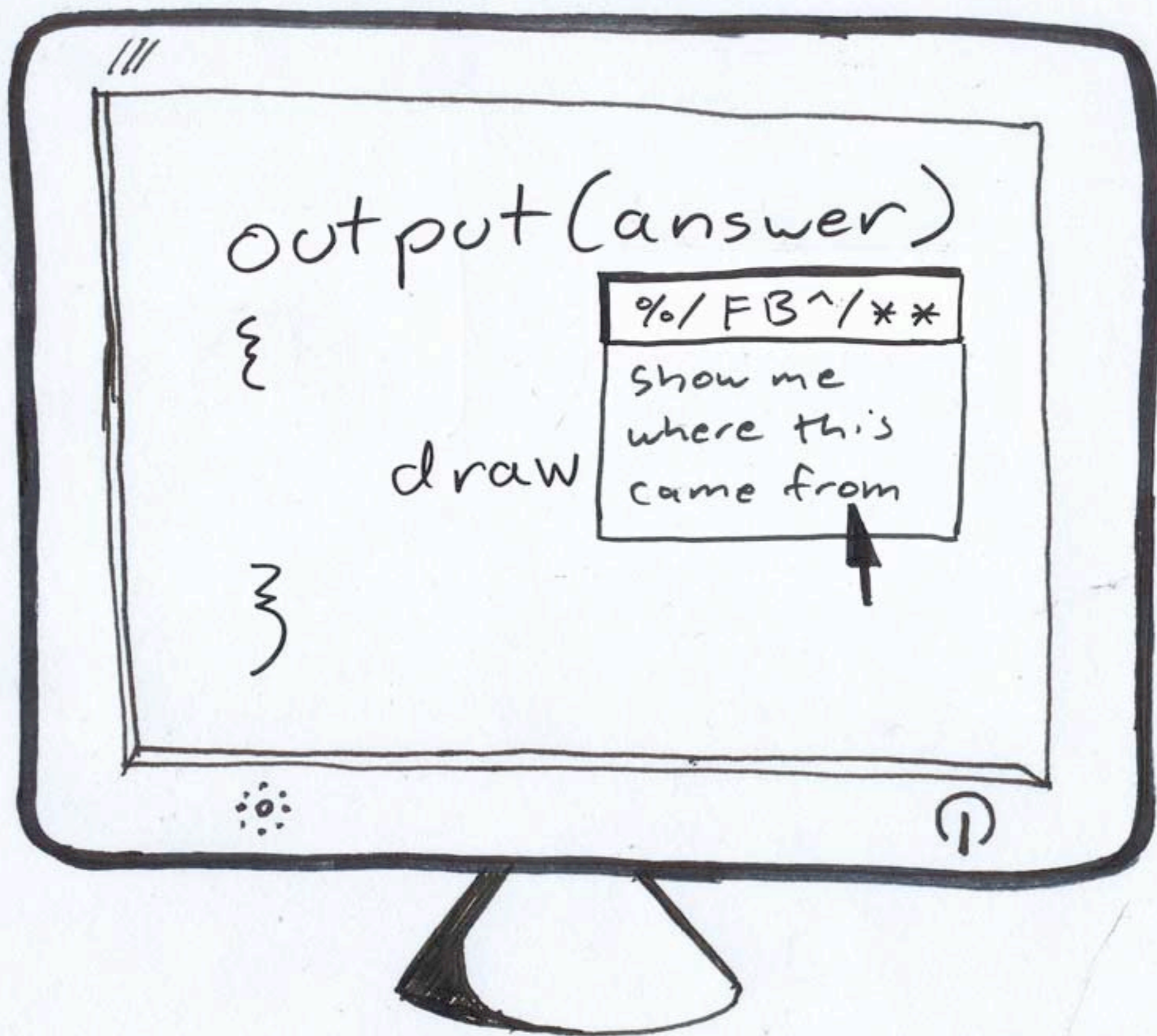












calculate()

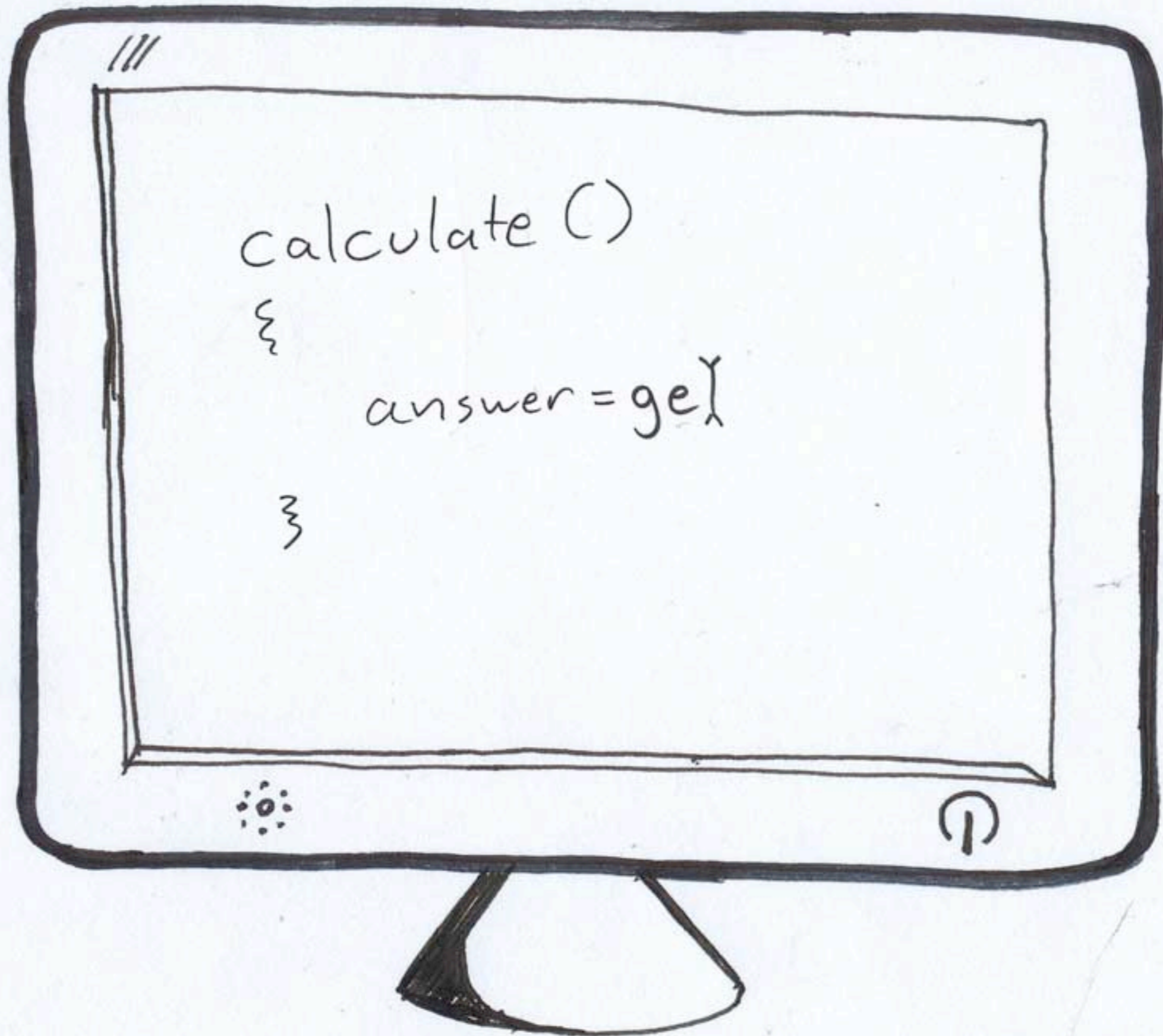
{

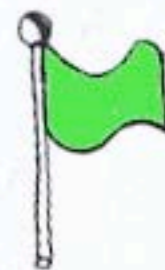
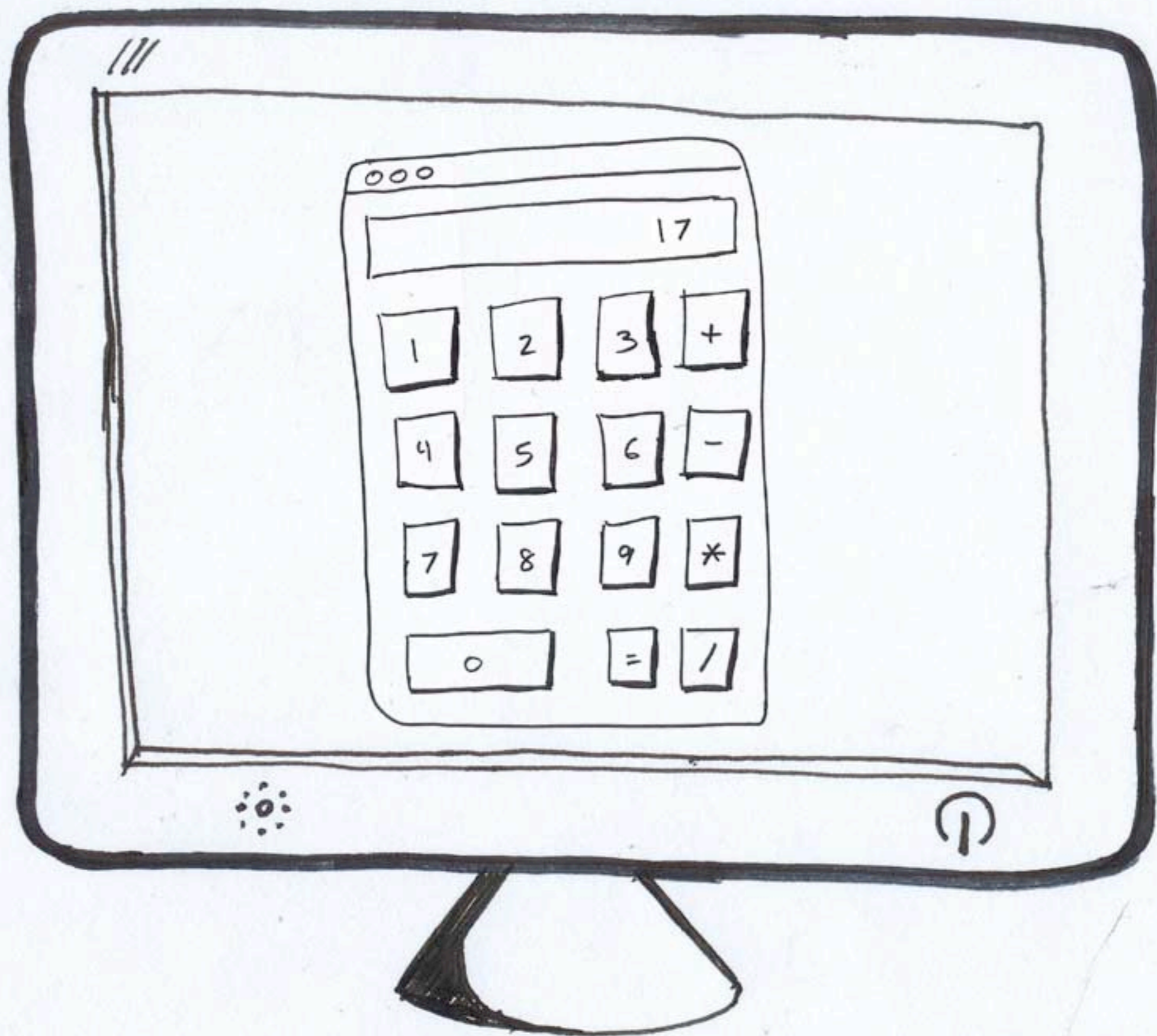
answer = "%/FIB~/\*\*";

}

Someone  
should  
probably  
fix this.







# Agenda

*context*

*problems today*

*visual metaphors*

*design goals*

*scenario from last review*

*interface specification*

*prototype for testing*

*impact*

*next steps*



- Graphics.java
- Physics.java
- Gameplay.java
- Compiler.java
- URL.java
- Compiler.java
- System.java
- Networking.java

```
class Graphics {  
    void start()  
    {  
        if (first_game)  
            loadEverything();  
        else  
            resetEverything();  
  
        for (int i = 0; i < shapes.length; i = i + 1)  
            draw(shapes[i]);  
    }  
}
```

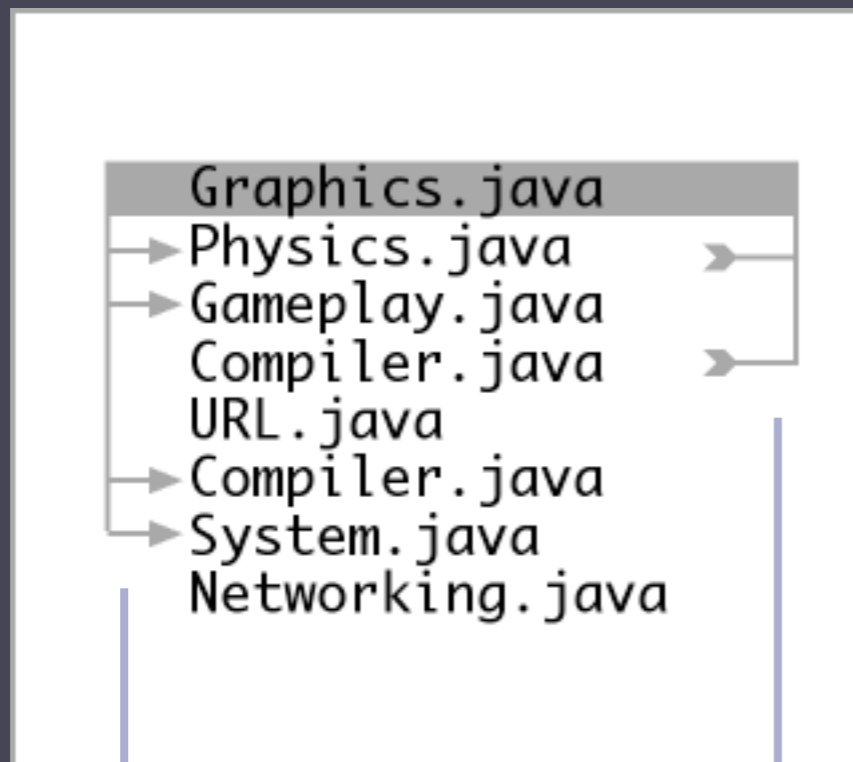
```
void draw(Shape shape <Rect:100x30px, (30, 25)> )  
{  
    print(shape.type Rectangle);  
    print("width: " + shape.width 100);  
    print("height: " + shape.height 30);  
    print(shape.vertices);  
    4:[<Vertex, Vertex, Vertex, Vertex>]  
}
```

```
void drawBorder(Shape shape)  
{  
}  
  
void drawBackground(Shape shape)  
{  
}  
  
void drawForeground(Shape shape)  
{  
}  
  
void loadEverything()  
{  
}  
}
```



main	
startup	
Graphics.start	
Graphics.loa...	22
Graphics.draw	3
Graphics.draw	3
Graphics.draw	3
Graphics.draw	3
Graphics.draw	3
Graphics.draw	3
Physics.start	5
Gameplay.start	123
run	3456
cleanup	291

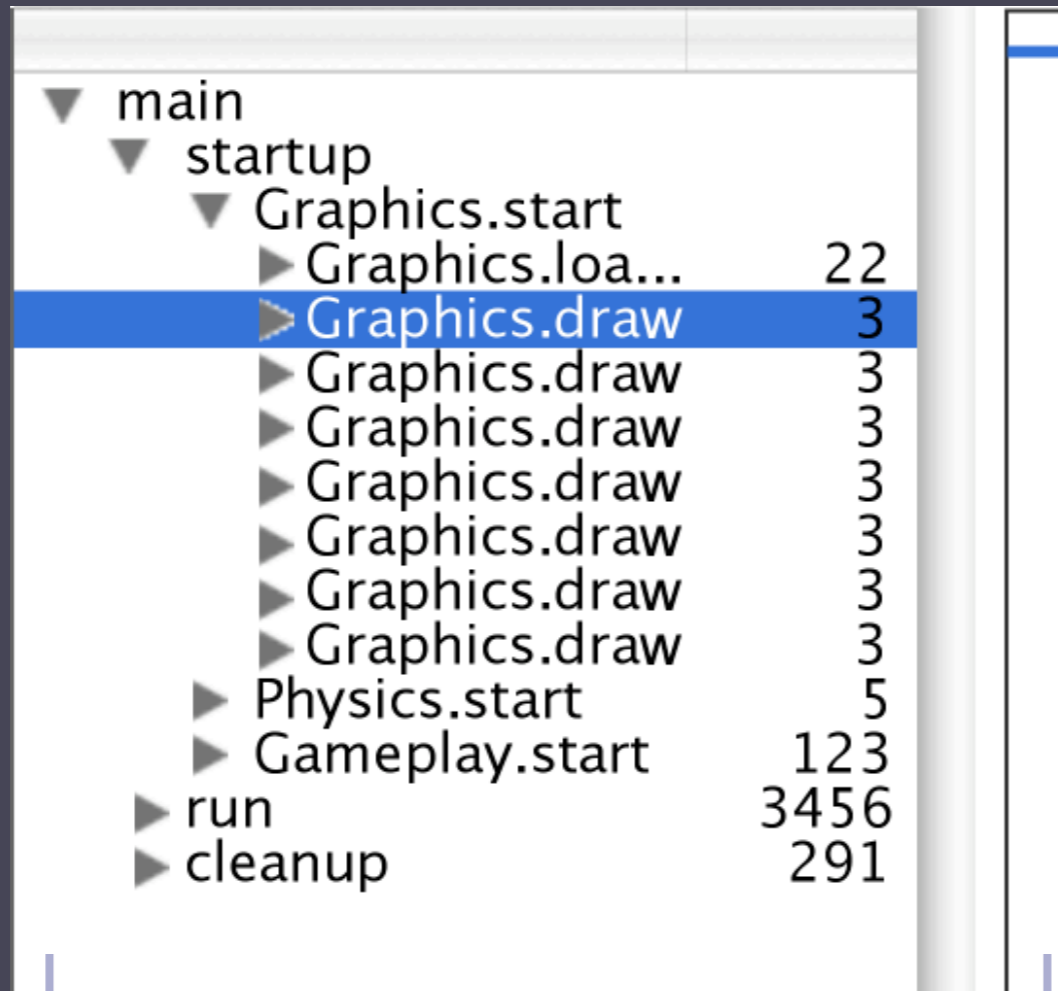
# Program files and dependencies



“Children”

“Parents”

# Timeline of program execution (“trace”)



The image shows a screenshot of a program execution trace. It is a hierarchical tree structure where each node represents a function call. The root node is 'main', which has a child 'startup'. 'startup' has a child 'Graphics.start', which in turn has several children: 'Graphics.loa...', 'Graphics.draw' (highlighted in blue), and several more 'Graphics.draw' calls. Below these are 'Physics.start', 'Gameplay.start', 'run', and 'cleanup'. To the right of each function name is a numerical value representing its execution time. The values are: main (22), startup (3), Graphics.start (3), Graphics.loa... (3), Graphics.draw (3), Graphics.draw (3), Graphics.draw (3), Graphics.draw (3), Graphics.draw (3), Graphics.draw (3), Physics.start (5), Gameplay.start (123), run (3456), and cleanup (291). The 'run' function has the highest execution time, followed by 'cleanup' and 'Gameplay.start'.

▼ main	
▼ startup	
▼ Graphics.start	
▶ Graphics.loa...	22
▶ Graphics.draw	3
▶ Graphics.draw	3
▶ Graphics.draw	3
▶ Graphics.draw	3
▶ Graphics.draw	3
▶ Graphics.draw	3
▶ Graphics.draw	3
▶ Physics.start	5
▶ Gameplay.start	123
▶ run	3456
▶ cleanup	291

Hierarchical time

Clock time



- Graphics.java
- Physics.java
- Gameplay.java
- Compiler.java
- URL.java
- Compiler.java
- System.java
- Networking.java

```
class Graphics {  
    void start()  
    {  
        if (first_game)  
            loadEverything();  
        else  
            resetEverything();  
  
        for (int i = 0; i < shapes.length; i = i + 1)  
            draw(shapes[i]);  
    }  
}
```

```
void draw(Shape shape <Rect:100x30px, (30, 25)> )  
{  
    print(shape.type Rectangle);  
    print("width: " + shape.width 100);  
    print("height: " + shape.height 30);  
    print(shape.vertices);  
    4:[<Vertex, Vertex, Vertex, Vertex>]  
}
```

```
void drawBorder(Shape shape)  
{  
}
```

```
void drawBackground(Shape shape)  
{  
}
```

```
void drawForeground(Shape shape)  
{  
}
```

```
void loadEverything()  
{  
}
```

```
}
```



- main
  - startup
    - Graphics.start
      - Graphics.loa... 22
      - Graphics.draw 3
      - Graphics.draw 3
      - Graphics.draw 3
      - Graphics.draw 3
      - Graphics.draw 3
      - Graphics.draw 3
      - Physics.start 5
      - Gameplay.start 123
    - run 3456
    - cleanup 291

# Live function: code operating on data

```
void draw(Shape shape <Rect:100x30px, (30, 25)>)
{
    print(shape.type Rectangle);
    print("width: " + shape.width 100);
    print("height: " + shape.height 30);
    print(shape.vertices);
    4:[<Vertex, Vertex, Vertex, Vertex>]
}
```

Complex value


Array value

Numerical value

# Live function: unreachable code

```
if (first_game true)  
    loadEverything();  
else  
    resetEverything();
```

# Live function: loops

```
i   for (int i = 0; i  < 10; i  = i  + 1)  
    print(i  );
```

# Live function hyperlinks

```
for (int i = 0; i 3 < shapes.leng  
    draw(shapes[i 3 ] );
```



```
void draw(Shape shape <Rect: 100x30px, (30, 25)> )  
{  
    drawBorder(shape <Rect: 100x30px, (30, 25)> ) ;  
    drawBackground(shape <Rect: 100x30px, (30, 25)> ) ;  
    drawForeground(shape <Rect: 100x30px, (30, 25)> ) ;  
}
```

Returns to calling function

# Flagging incorrect values

```
print("width: " + shape.width 100 );  
print("height: " + shape.height);  
print(shape.vertices);
```

Previous change  
Find in trace  
Flag

↓

```
print("width: " + shape.width 100 correct value );
```

↓

```
print("width: " + shape.width 100 25 );
```



- Graphics.java
- Physics.java
- Gameplay.java
- Compiler.java
- URL.java
- Compiler.java
- System.java
- Networking.java

```
class Graphics {  
    void start()  
    {  
        if (first_game)  
            loadEverything();  
        else  
            resetEverything();  
  
        for (int i = 0; i < shapes.length; i = i + 1)  
            draw(shapes[i]);  
    }  
}
```

```
void draw(Shape shape <Rect:100x30px, (30, 25)> )  
{  
    print(shape.type Rectangle);  
    print("width: " + shape.width 100 correct value );  
    print("height: " + shape.height 30);  
    print(shape.vertices);  
    4:[<Vertex, Vertex, Vertex, Vertex>]  
}
```

```
void drawBorder(Shape shape)  
{  
}
```

```
void drawBackground(Shape shape)  
{  
}
```

```
void drawForeground(Shape shape)  
{  
}
```

```
void loadEverything()  
{  
}
```

```
}
```

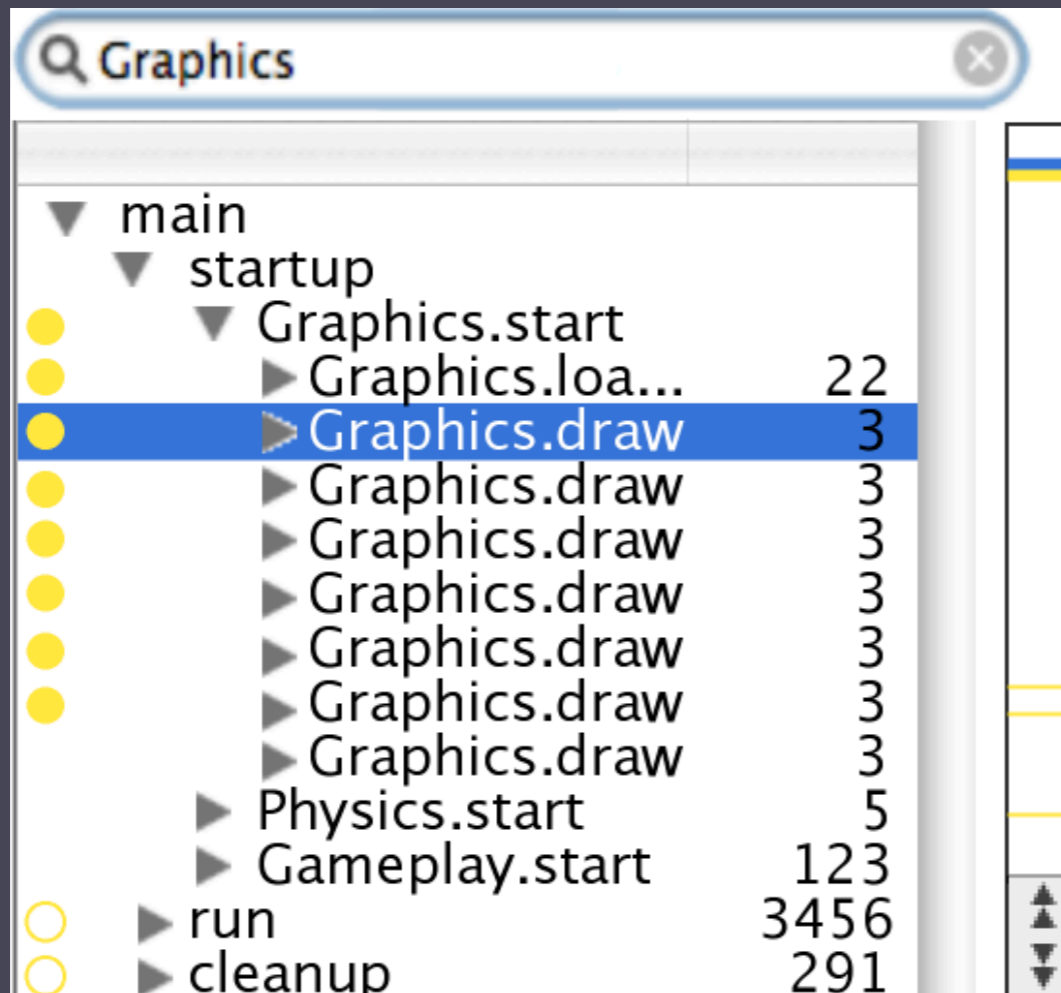


- main
  - startup
    - Graphics.start
      - Graphics.loa... 22
      - Graphics.draw 3
      - Graphics.draw 3
      - Graphics.draw 3
      - Graphics.draw 3
      - Graphics.draw 3
      - Graphics.draw 3
      - Physics.start 5
      - Gameplay.start 123
    - run 3456
    - cleanup 291

# Flags in the program timeline

▼	main	
▼	startup	
▼	Graphics.start	
▶	Graphics.loa...	22
▶	Graphics.draw	3
▶	Graphics.draw	3
▶	Graphics.draw	3
▶	Graphics.draw	3
▶	Graphics.draw	3
▶	Graphics.draw	3
▶	Physics.start	5
▶	Gameplay.start	123
▶	run	3456
▶	cleanup	291

# Search: file, function, variable



Graphics		
▼	main	
▼	startup	
●	▼ Graphics.start	
●	▶ Graphics.loa...	22
●	▶ Graphics.draw	3
●	▶ Graphics.draw	3
●	▶ Graphics.draw	3
●	▶ Graphics.draw	3
●	▶ Graphics.draw	3
●	▶ Graphics.draw	3
	▶ Physics.start	5
	▶ Gameplay.start	123
○	▶ run	3456
○	▶ cleanup	291

# Variable traces

x			
	▼	main	
	▼	startup	
100	▼	Graphics.start	
32	▶	Graphics.loa...	22
24	▶	Graphics.draw	3
	▶	Graphics.draw	3
	▶	Graphics.draw	3
	▶	Graphics.draw	3
	▶	Graphics.draw	3
	▶	Graphics.draw	3
	▶	Graphics.draw	3
8	▶	Physics.start	5
	▶	Gameplay.start	123
	▶	run	3456
	▶	cleanup	291

## Programmer feedback

Early draft of wireframes.

Reviewed by 7 professional programmers.

Approval of general approach and structure.

Requests for specific features.

Many of which have been incorporated.

# Agenda

*context*

*problems today*

*visual metaphors*

*design goals*

*scenario from last review*

*interface specification*

*prototype for testing*

*impact*

*next steps*

# Agenda

*context*

*problems today*

*visual metaphors*

*design goals*

*scenario from last review*

*interface specification*

*prototype for testing*

*impact*

*next steps*

# Impact

More productive programmers.

Better and more complex software.

# Agenda

*context*

*problems today*

*visual metaphors*

*design goals*

*scenario from last review*

*interface specification*

*prototype for testing*

*impact*

*next steps*

## Next steps

User testing and validation.

Complete interface specification.

Exhibition preparation.

Thank you.