UC Wellbeing | Oranga

UC UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha

How to PROTECT YOURSELF and others

Stay home if you feel unwell and call Healthline on

0800 358 5453

- Cough or sneeze into elbow or by covering mouth and nose with tissues.
- Put used tissues in the bin or a bag immediately.
- Wash your hands with soap and water often.
- Try to avoid close contact with people who are unwell.
- Don’t touch your eyes, nose or mouth if your hands are not clean.
- Clean and disinfect frequently touched surfaces and objects, such as doorknobs.
- Look at scientific facts, not hype. Be kind to one another.

User Interface Design

ENCE361: Lecture 13

Roadmap

		ENCE361 Lecture, Tutorial & Lab Schedule – 2020						v. 20.1	Updated 13/02/2020
		Lec 1		Tute	Lec 2			Lec 3	Lab
Wk	Starting	Mon 1p		Tue 11a, Wed 8a	Wed 2p			Fri 2p	Mon 11a, Tue 9a, Tue 11a, Wed 11a
1	17 Feb	1	Intro; computer arch.	No tute	2	Introduction to ARM	3	General purpose I/O	Lab 0 – Use of Test Equipment (non-270 only)
2	24 Feb	4	Signals; data acq.	Use of CCS	5	Electrical noise	6	A-to-D conversion	Lab 1 – CCS & GPIO
3	2 Mar	7	Interrupts (systick)	Labs help	8	Buffers	9	I2C/SPI	Lab 2 – Accelerometer & buttons
4	9 Mar	10	Switch debouncing (FSM 1)	Project intro	11	FFs, counters, etc.	12	Timers	Lab 3 – Data acquisition & interrupts
5	16 Mar	13	UI design (FSM 3)	Source control with Git	14	Fore-/background	15	Interrupt latency	**Project – Milestone 1: Accelerometer & OLED**
6	23 Mar	16	Recap on architecture	Homework help	17	Atomicity, sharing	18	Quadrature decoding (FSM 2)	Project
7	30 Mar	19	PID	Test Prep	20	Serial ports	21	Revision	Project; **WED or THU (tbc): TEST (L1–L18)**
			3 week term break						
8	27 April	22	ANZAC Day holiday	Test Answers	23	Kernels 1	24	Kernels 2	**Project – Milestone 2: GUI**
9	4 May	25	CPU load analysis	Homework help	26	Microcontroller interfacing	27	Kernels 3	Project
10	11 May	28	Memory structures	Homework help	29	MCU memory types	30	Arm Arithm./Logic ccts	Project
11	18 May	31	The ARM ISA	Homework help	32	ARM Assembly language	33	IEEE FP representation	**Project demos in usual lab slot**
12	25 May	34	Revision (LY)						
13	1 June			Revision/Exam Prep (CM)					**Project report & code due Fri 29 May**
	Key:	Le Yang							
		Ciaran Moore							

Ciaran Moore
ciaran.moore@canterbury.ac.nz

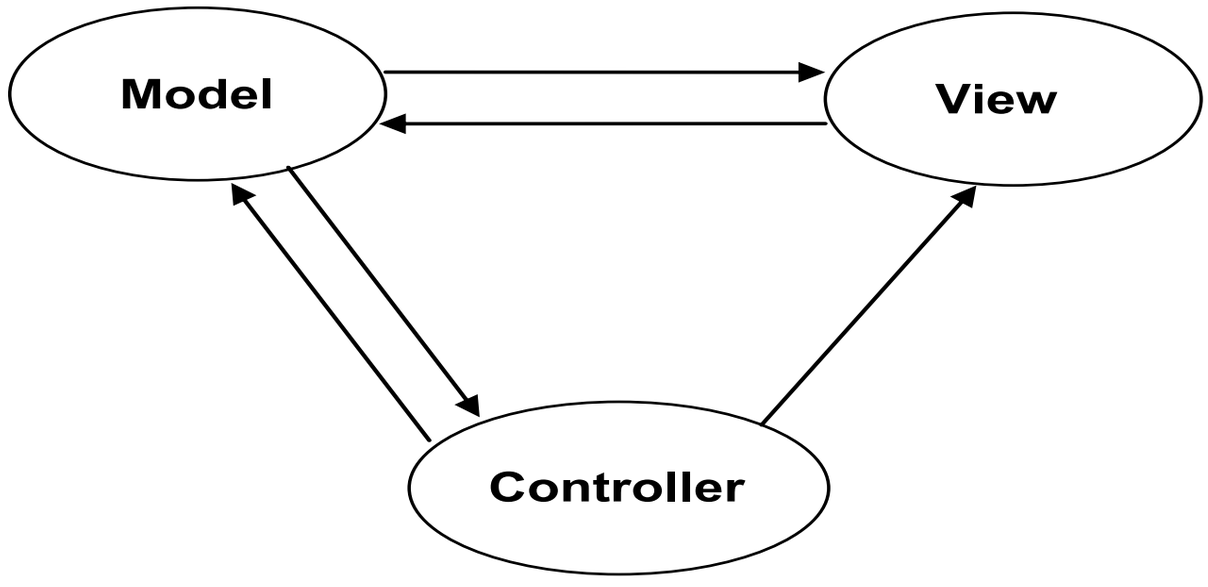
Outline

- Model View Controller Architecture
- Aspects of GUI Design

Ciaran Moore
ciaran.moore@canterbury.ac.nz

MVC Architecture

- Model View Controller architecture is a way of representing window management systems for GUIs. It is also useful for simpler UIs. It comprises three interacting subsystems:



Note: these are components or modules, *not* states.

Ciaran Moore
ciaran.moore@canterbury.ac.nz

Model

- The Model subsystem maintains a database of the UI's current status: information such as where a flashing cursor is located, the size and location of a clickable area, the current font and/or size of a graphical object.
- The Model represents the **state** of the UI.

Ciaran Moore
ciaran.moore@canterbury.ac.nz

Model

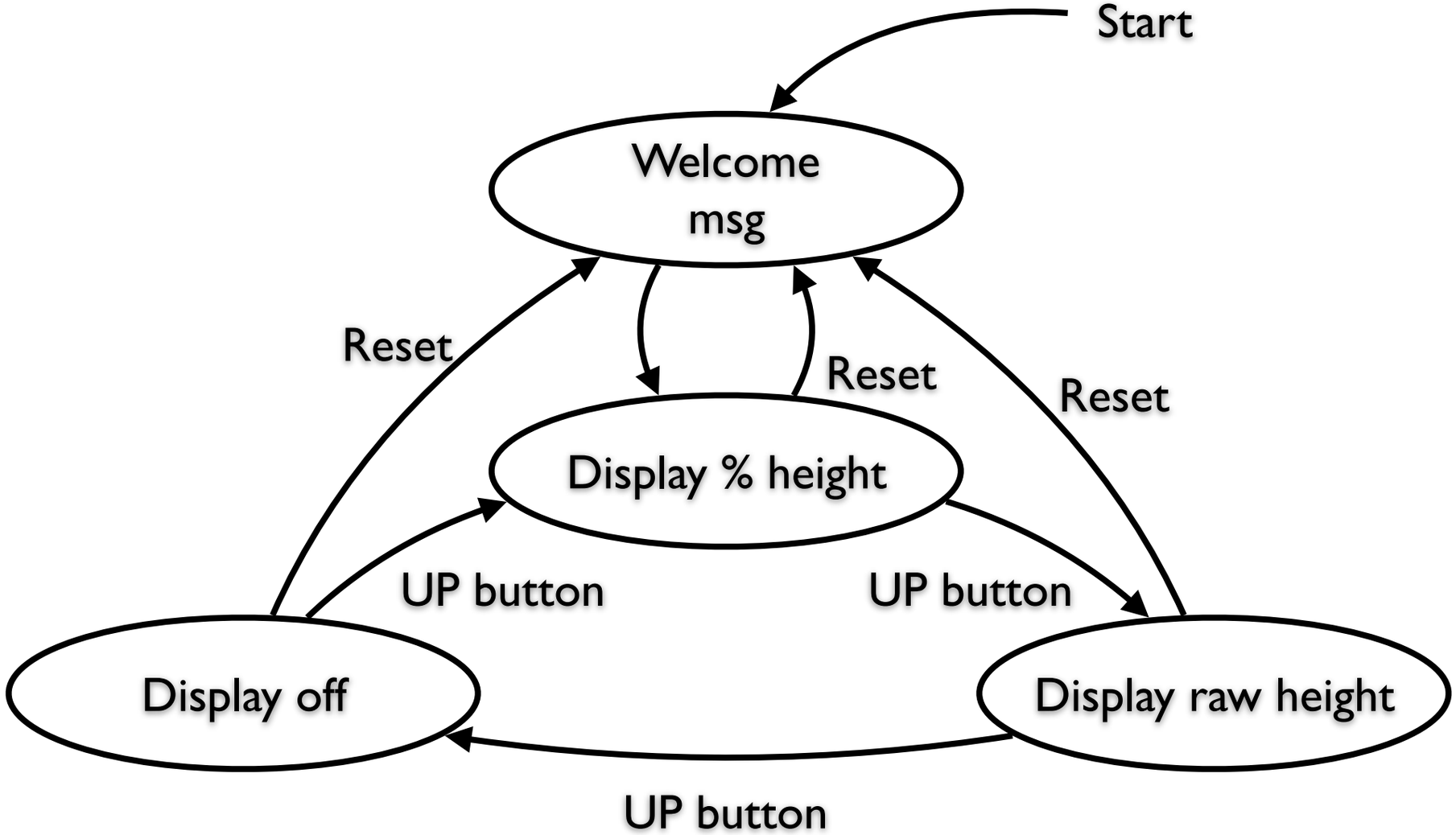
- Examples of Model components:
  - A function which changes the state of the program, e.g. changing from displaying height as a percentage to raw ADC values.
  - A function which is called when a button is physically pushed and uses the information from the identity of the key to change the state of the model.
  - A structure which maintains knowledge of the buttons.

```
// Code from MS1:
switch (displayMode)
{
   case PERCENT:
     updateDisplay (height_pct);
     break;
  case RAW:
     updateDisplay (height_raw);
     break;
  case OFF:
     updateDisplay (0);
}
```


Ciaran Moore
ciaran.moore@canterbury.ac.nz

Model

- Often the Model subsystem can be thought of as a *finite state machine*.



View

- The View subsystem provides the conversion of information stored in the Model subsystem to a graphical display, i.e. a set of functions which generate the visual appearance of the display.
- It is invoked by the controller subsystem responding to application signals (e.g. calls to an `update` function).

Ciaran Moore
ciaran.moore@canterbury.ac.nz

View

- Examples of View components:
  - A set of global constants which define the way option prompts appear on the display.
  - The principal `draw()` function.
  - A set of supplementary drawing functions, including those which put particular patterns and shapes on the display.
  - The functions with prototypes in the `lib_OrbitOled` set.

```
updateDisplay (height, percent);
OLEDStringDraw (string, 0, 2);
```


Ciaran Moore
ciaran.moore@canterbury.ac.nz

Controller

- The controller subsystem provides the interface between the events (e.g. external signal, button pushed, etc.) and the application program's responses. It controls which *event handler function*, one for each type of event, gets called; those functions will update the contents of the model.
- In turn, the event handlers need a mechanism to 'broadcast' that something has occurred which requires the visual appearance of the UI to change; this mechanism is the call to the `redisplay()` function.

Ciaran Moore
ciaran.moore@canterbury.ac.nz

Controller

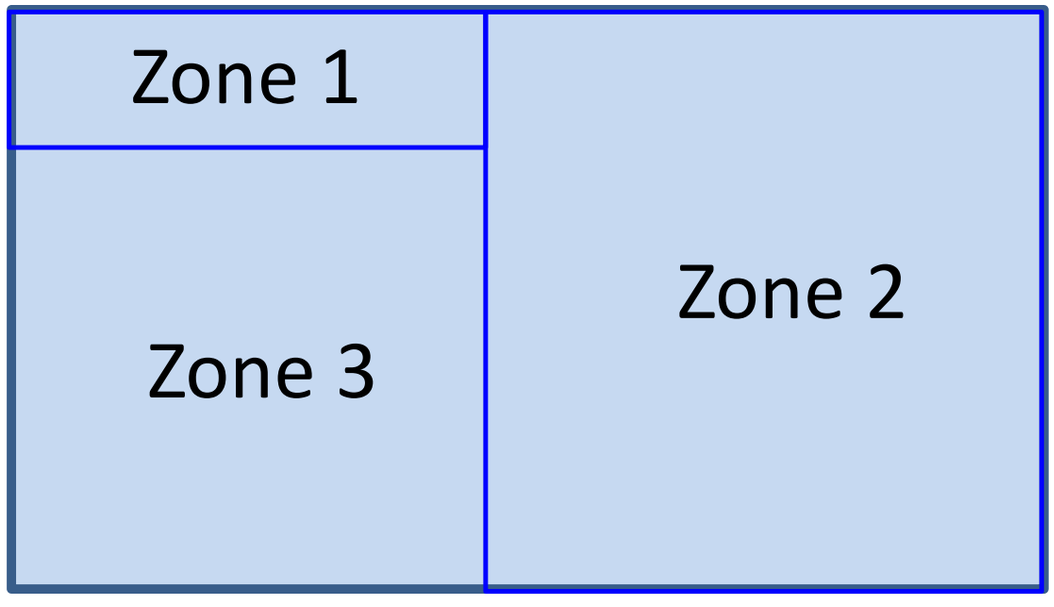
- Examples of Controller components:
  - An initialisation function which sets up the display.
  - The `redisplay()` function which enables the Model subsystem to signal the need for changes to appear in the visual appearance of the zone of the display.

```
void buttonPushed (void)
{
  // Change height disp. setting:
buttonState = checkButton (UP);
if (buttonState == PUSHED)
{
  displayMode++;
  if (displayMode > OFF)
    displayMode = PERCENT;
  }
}
}
```


Ciaran Moore
ciaran.moore@canterbury.ac.nz

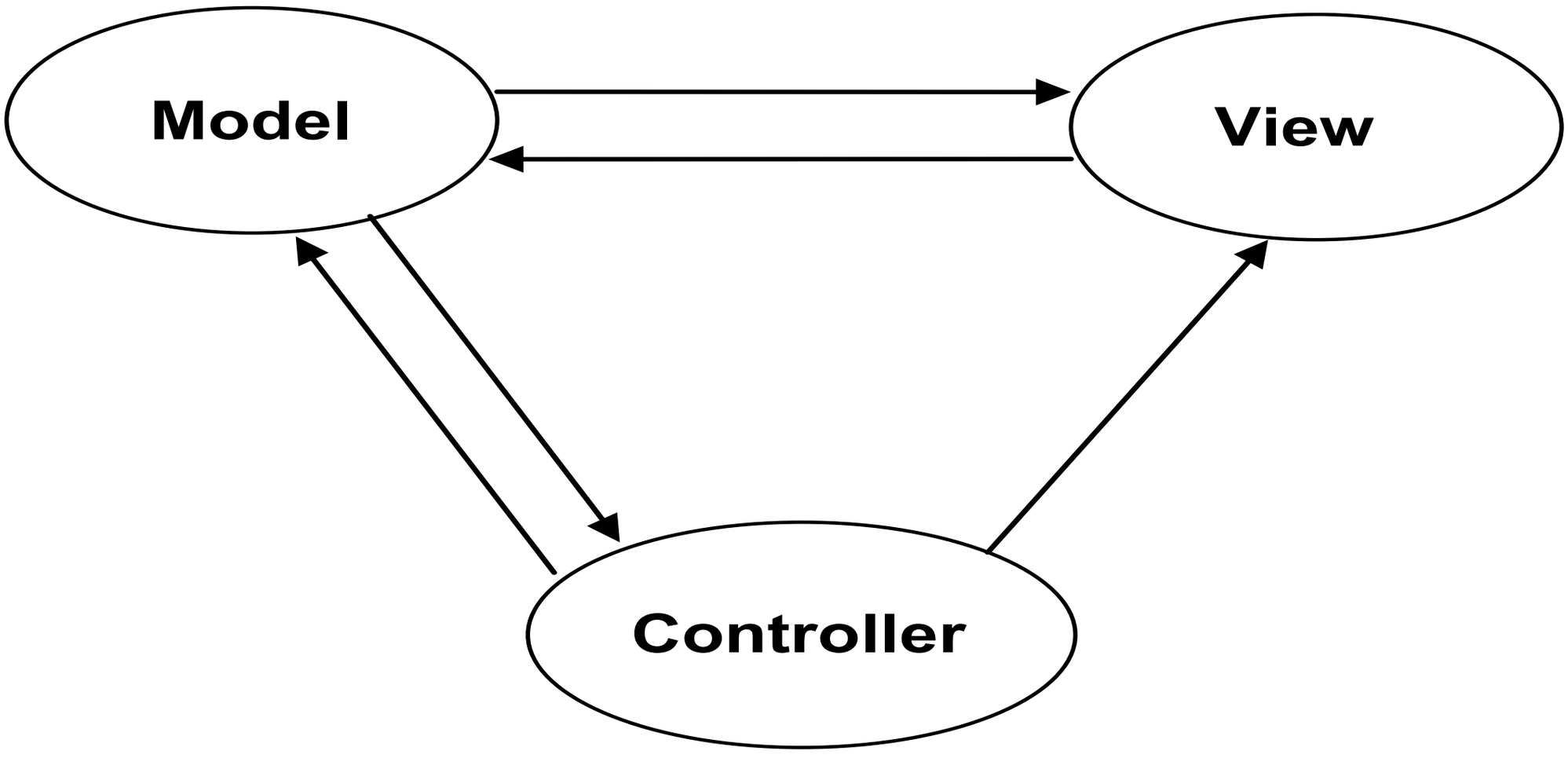
Controller

- Often clearing and redrawing the whole display can be slow: it can be better to just update the part of the display that needs changing.
- One approach is to divide the display into zones, each identified by a name.
- So the call to the controller function changes from e.g. `redisplay()` to `redisplay(zone_1)`.



Ciaran Moore
ciaran.moore@canterbury.ac.nz

MVC Architecture



Ciaran Moore
ciaran.moore@canterbury.ac.nz

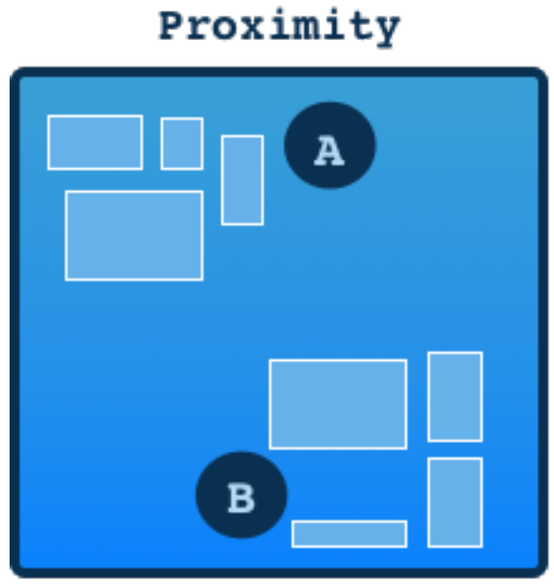
Aspects of GUI Design

- Layout
  - Gestalt Principles
- Navigation
- Terminology

Ciaran Moore
ciaran.moore@canterbury.ac.nz

Layout

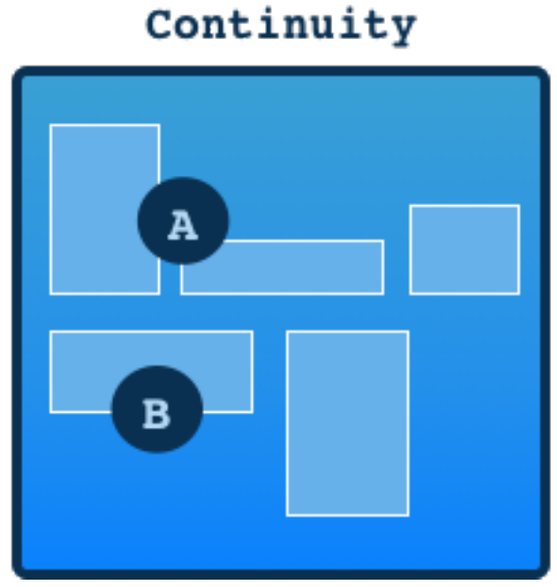
- Gestalt Principles:
  - Proximity – users assume that elements that are placed close together are related.

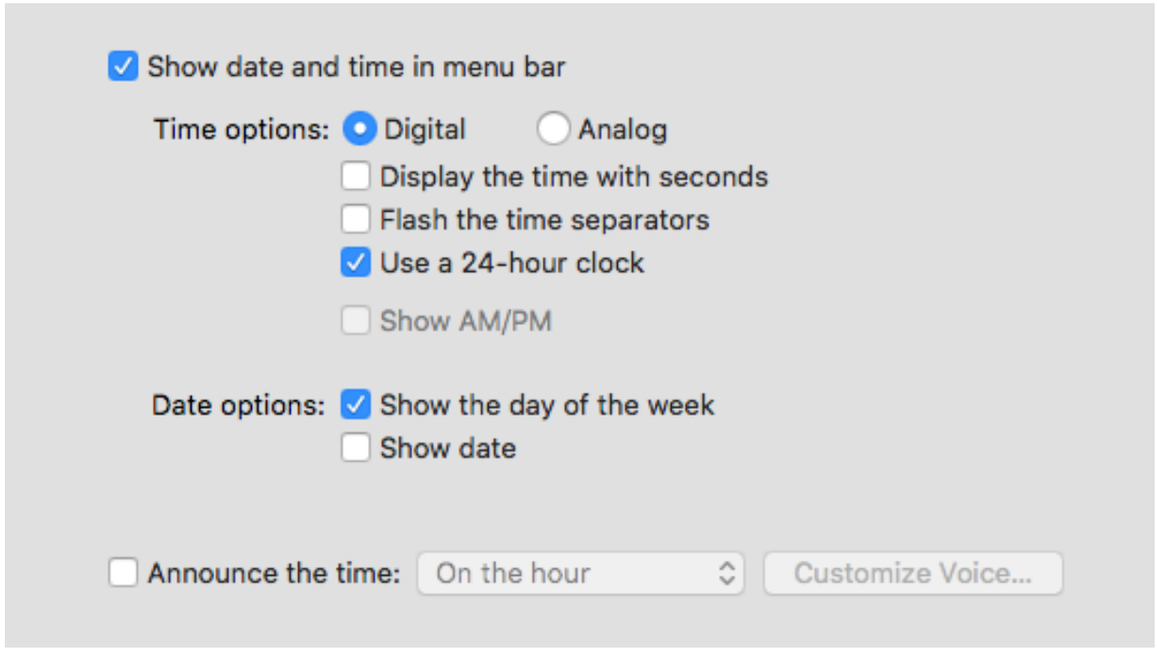


https://medium.com/@Jernfrost/understanding-visual-layout-in-gui-design-2fa91792edef

Layout

- Gestalt Principles:
  - Continuity – users assume that aligned objects are related.

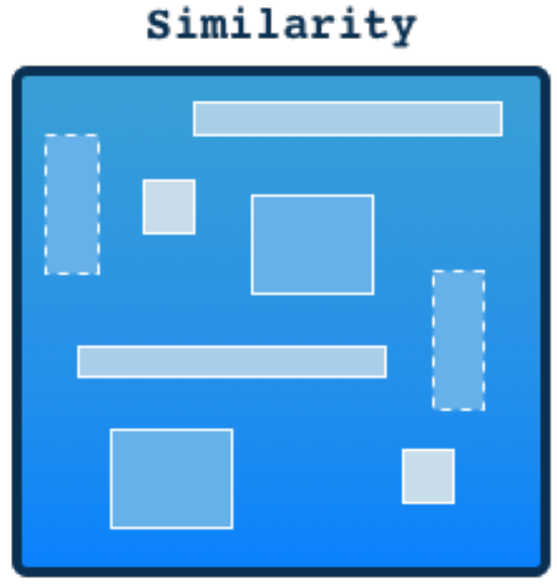


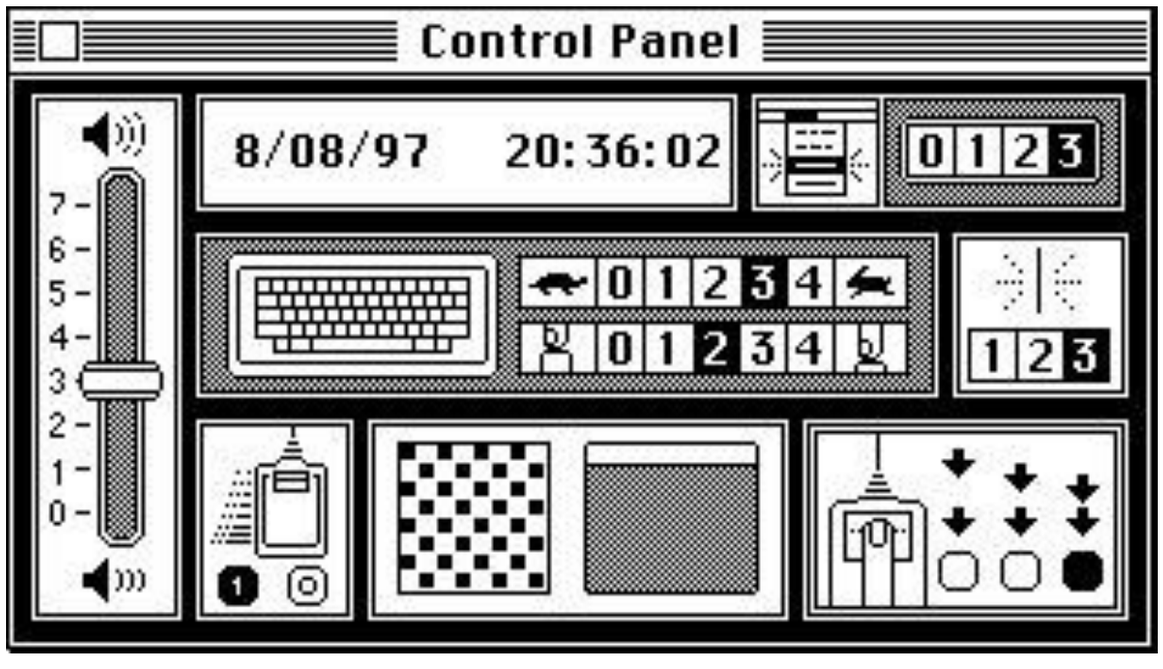


https://medium.com/@Jernfrost/understanding-visual-layout-in-gui-design-2fa91792edef

Layout

- Gestalt Principles:
  - Similarity – users assume that objects of similar size, colour and shape are related.

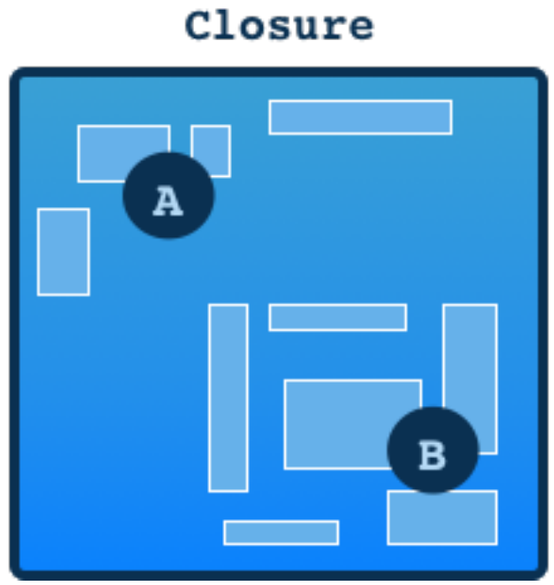


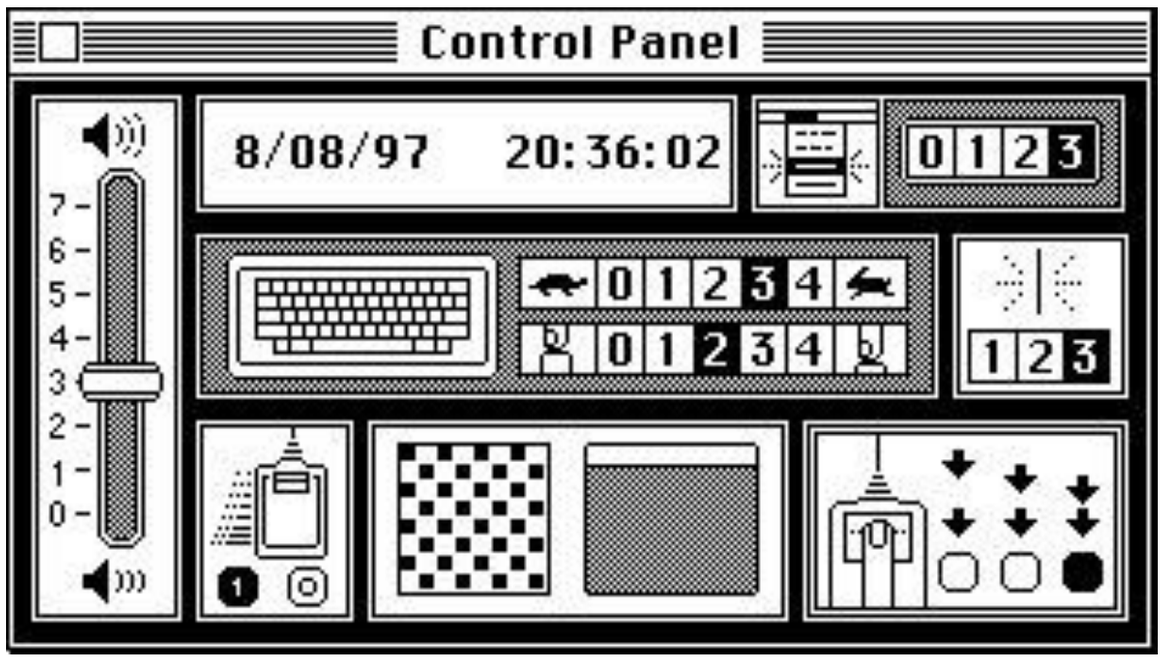


https://medium.com/@Jernfrost/understanding-visual-layout-in-gui-design-2fa91792edef

Layout

- Gestalt Principles:
  - Closure – users naturally try to enclose objects if they form a boundary.
    We can use this to convey that objects within an enclosure are part of a group.





https://medium.com/@Jernfrost/understanding-visual-layout-in-gui-design-2fa91792edef

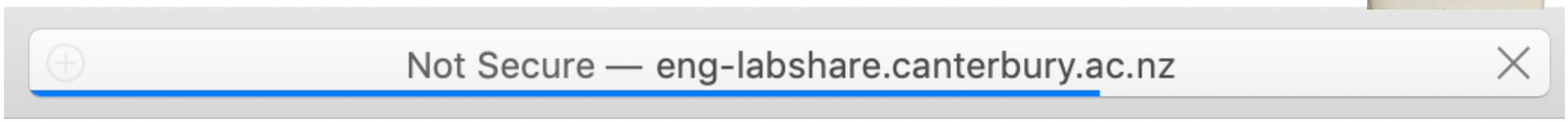
Navigation

- Following the gestalt principles when designing GUI elements helps users to navigate through series of sequential steps.
- Help the user further by providing an obvious starting point in the window's upper-left corner.
- Locate command buttons at the end of the sequence (bottom right of the window).

Ciaran Moore
ciaran.moore@canterbury.ac.nz

Navigation

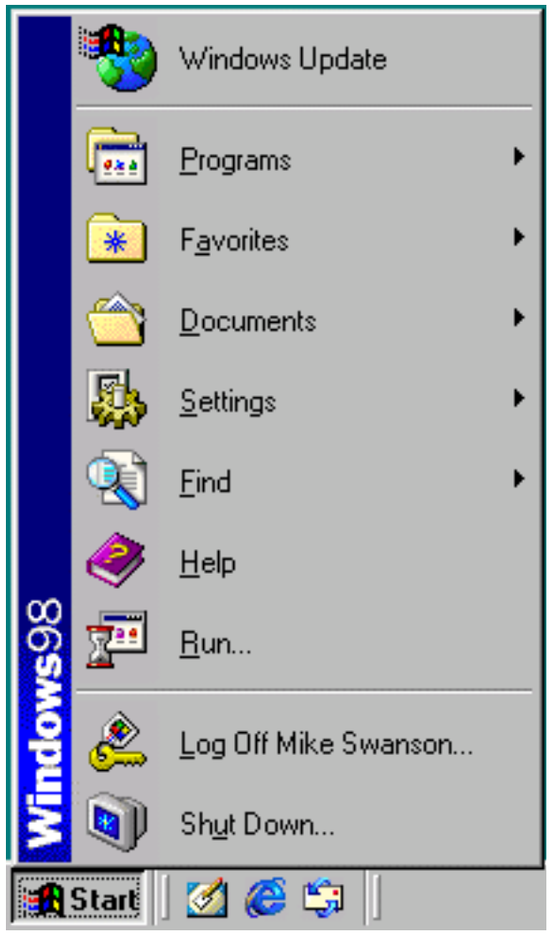
- Good UI acknowledges user actions
  - There is nothing more frustrating that clicking or tapping a button and getting no or slow response. Do you tap again?



Ciaran Moore
ciaran.moore@canterbury.ac.nz

Terminology

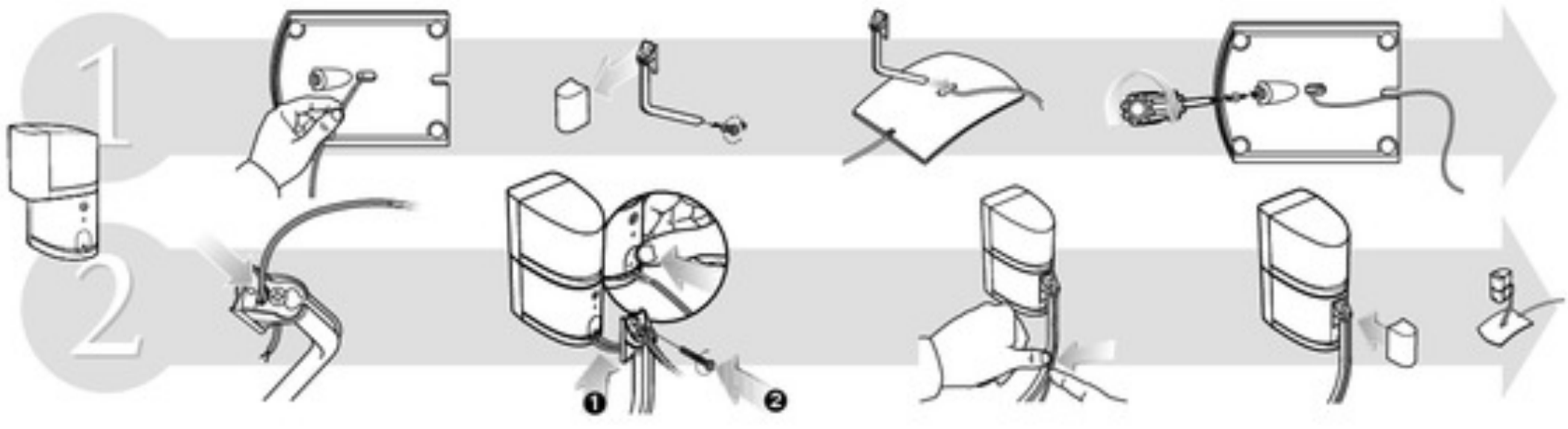
- Users hate inconsistent terminology in UIs.
- Ideally, use one word per concept:
  - e.g. choose one of find, search, query, inquiry and stick to it.
- Similarly, use one concept per word:
  - Be careful with words like *view* and *format*, which can be both verbs and nouns.
- Remember to be consistent across all aspects of your software:
  - Design documentation
  - Comments in code
  - variable, file & function names



Ciaran Moore
ciaran.moore@canterbury.ac.nz

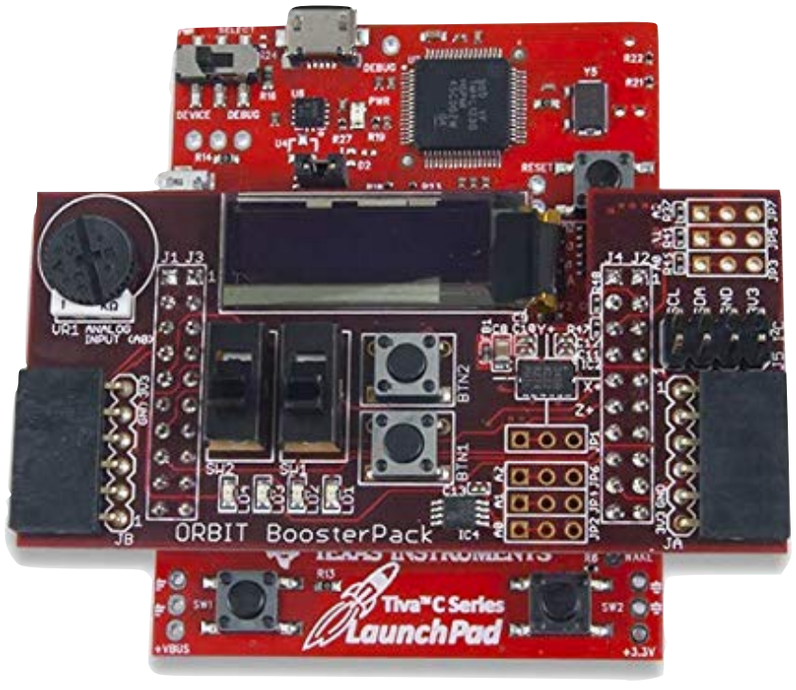
Terminology

- A sensible test for good GUI design: Can all display elements be identified by cues other than by reading the words that make them up?

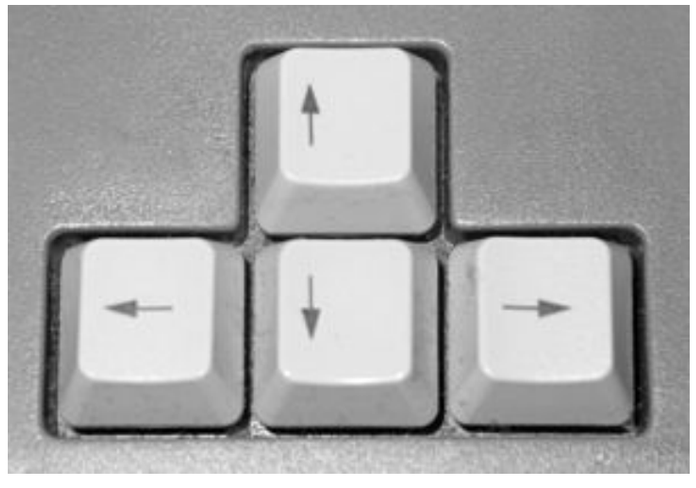


GUI Design for Embedded Systems

- Limited space and limited hardware make UI design for embedded systems especially hard.

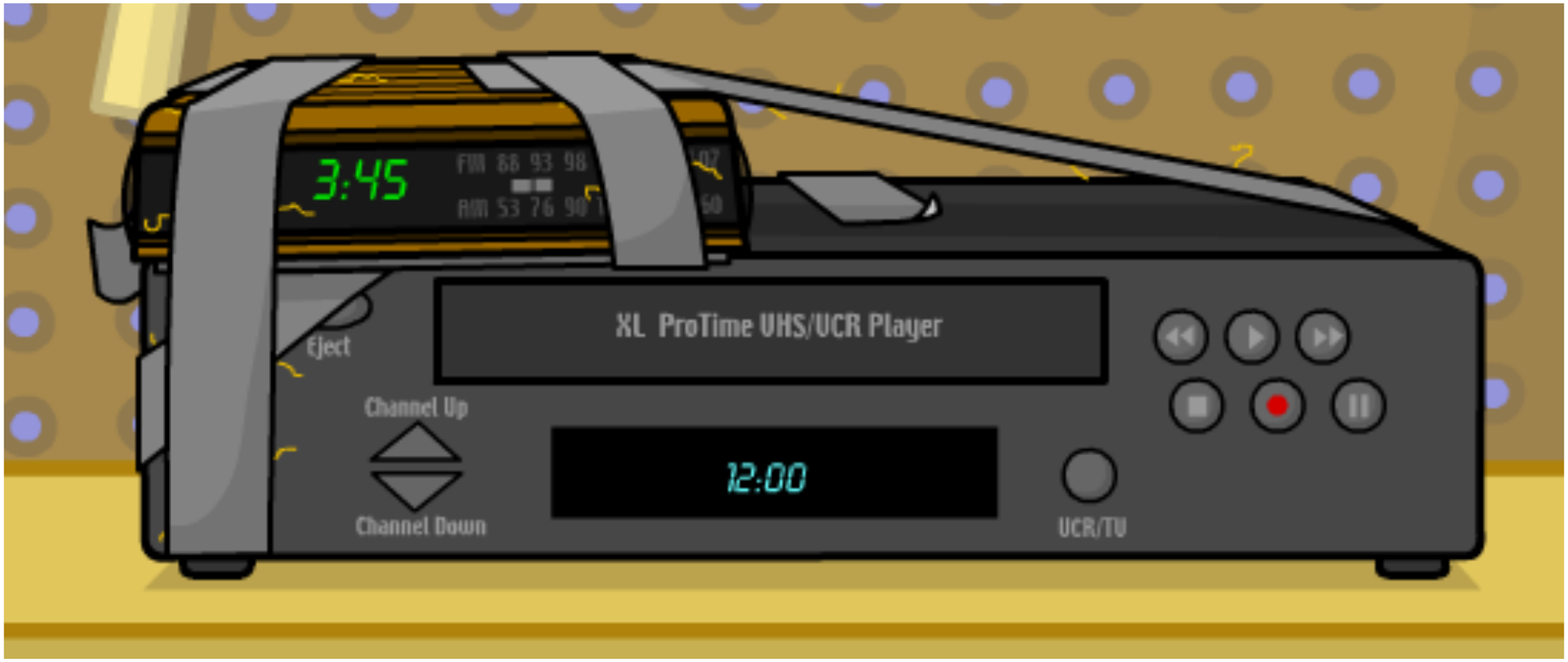


vs.



Ciaran Moore
ciaran.moore@canterbury.ac.nz

GUI Design for Embedded Systems



Ciaran Moore
ciaran.moore@canterbury.ac.nz

Homework

- Consider a GUI which implements the game Tetris on the TiVA using the navigation buttons (e.g. UP to rotate the falling piece clockwise and slider SW1 for starting a new game). List a set of functions that would be needed for the GUI and decide which would belong in each of the MVC components.
- Design an FSM (in the form of a state transition diagram) which could be used to control a simple oscilloscope-type display on the TiVA (e.g. setting vertical and horizontal scale, selecting input channel, etc.). Assume that, as well as support for writing characters (which you have already), support is available for graphics (via individual pixel addressing) on the 128 × 21 OLED display.
- With the aid of diagrams, explain the concepts of *proximity*, *continuity*, *similarity*, and *closure* as they relate to graphical user interfaces.

Ciaran Moore
ciaran.moore@canterbury.ac.nz

References

- Johnson, J. *"GUI Bloopers: Don't's and Do's for Software Developers and Web Designers"*, Morgan Kaufman, 2000.
- Galitz, W.O. *"The Essential Guide to User Interface Design"*, Wiley, 1996.
- Engheim, E. *"Understanding Visual Layout in GUI Design"*, Medium.com, 2017.