CS/INFO 4154:

Analytics-driven Game Design

Class 7:

User Interfaces

Mon	Wed	Fri
		9/8 Learnability Part 2
9/11 Learnability Part 3	9/13 Learnability Part 4	9/15 Throwaway Testing 1
9/18 Throwaway Testing 2		
	9/27 Alpha Testing 1	9/29 Alpha Testing 2

Assignment 5: Throwaway Prototype

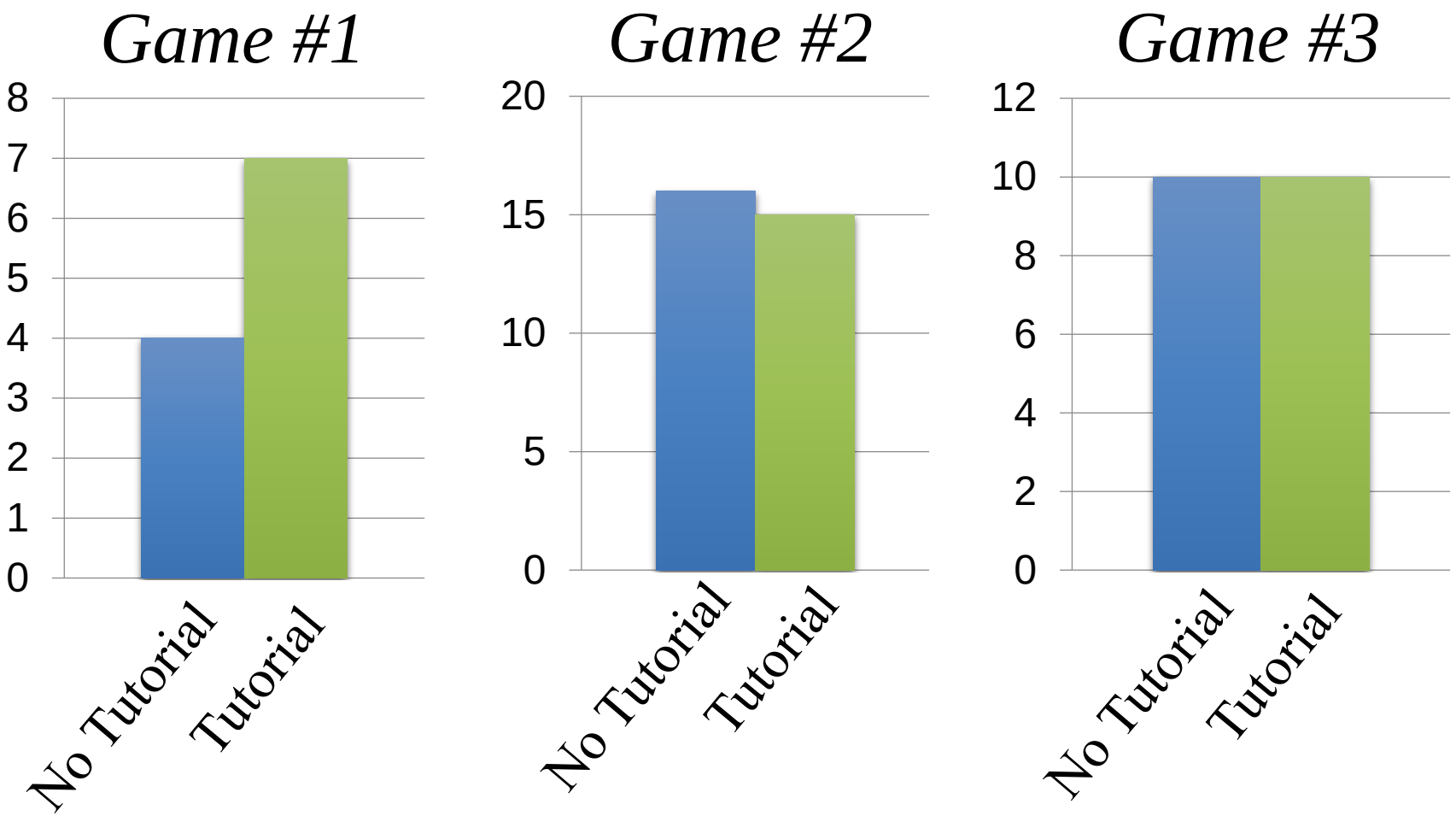
- **No pressure**
- Doesn’t need to be playable or integrated
- Pick *some pieces* of your game and build them
  - Avatar moves/jumps on flat land
  - Grid with nothing on it
  - Background artwork
- Submit picture through CMS by *end of class* on Friday 8/15

Review: Learnability

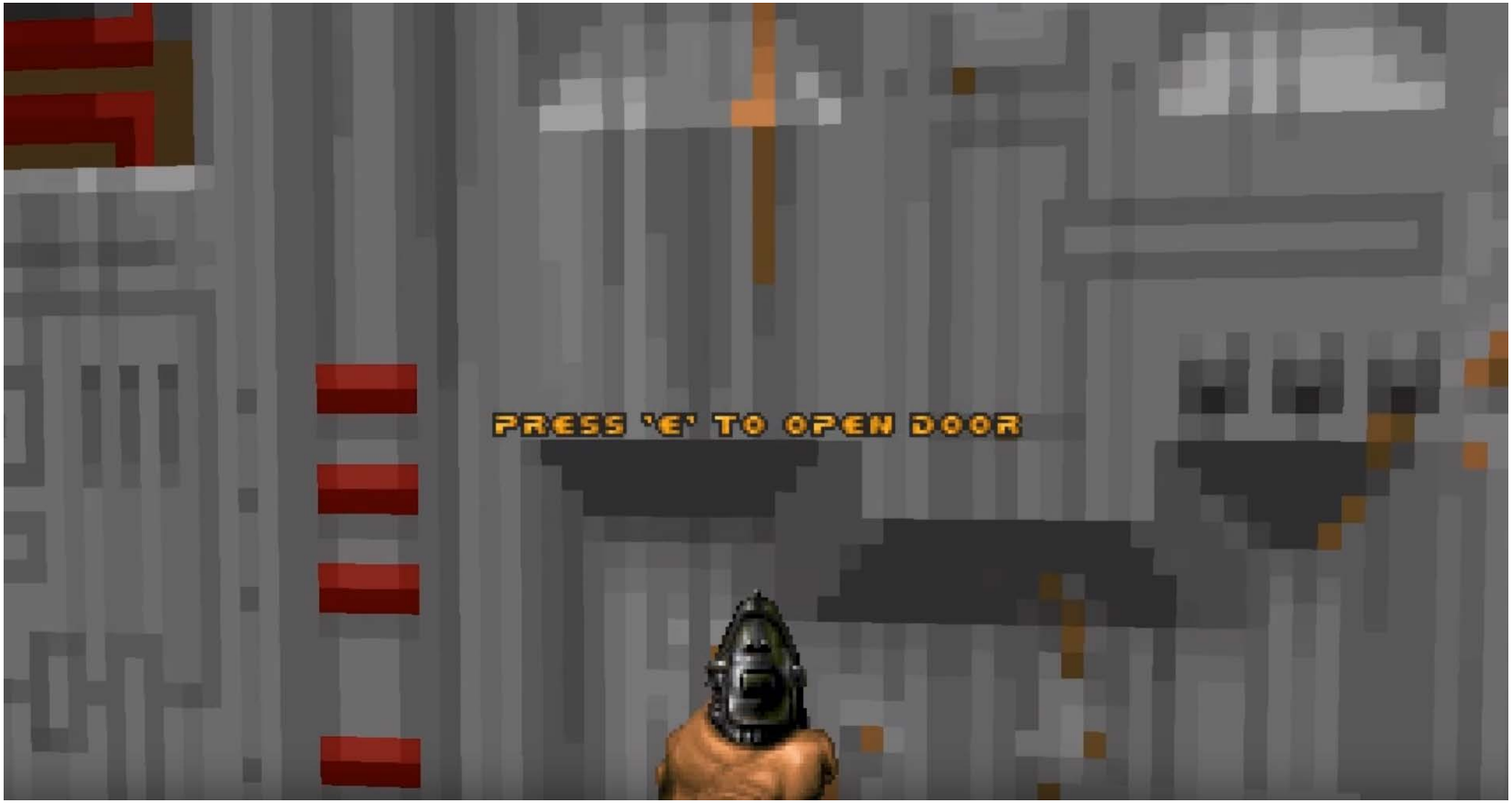
Nobody reads and ***nobody listens***

Review: Tutorials

- Tutorials have questionable effectiveness



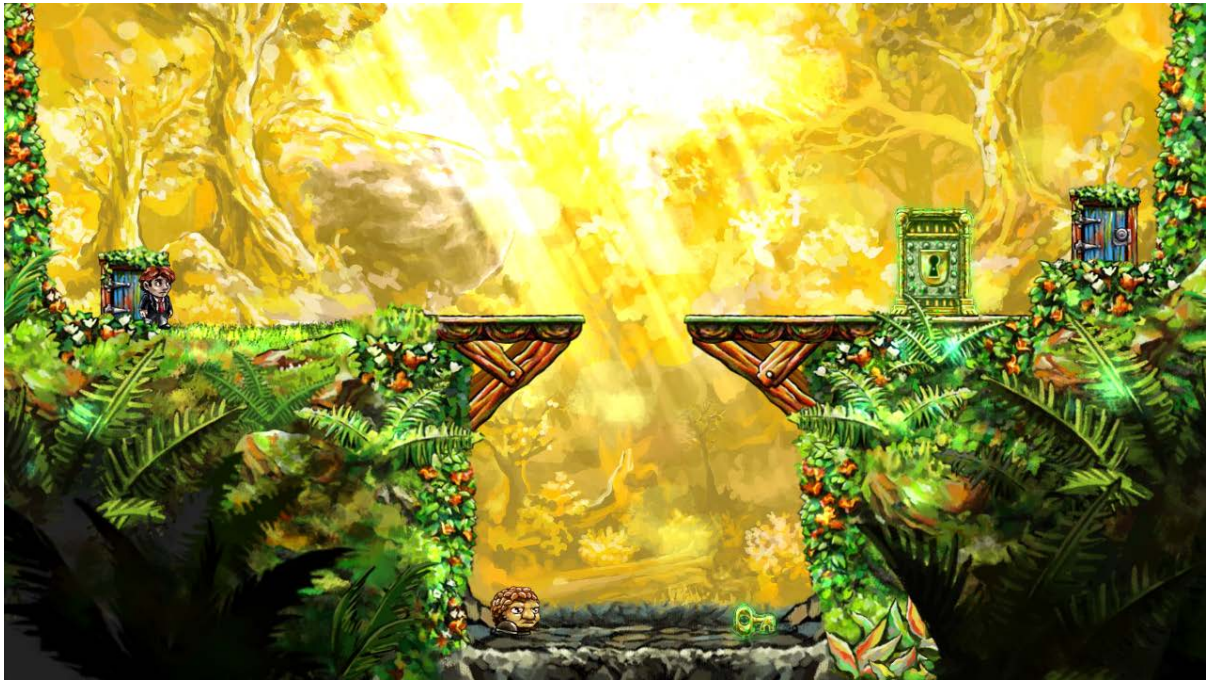
Review: *teach* actions



Review: *discover* interactions

Design ***discoverable situations***:

- *Impossible to pass* without experiencing interaction
- *Isolated* from other actions and interactions
- Player is relatively *safe*



Now: User Interfaces

Elder Scrolls IV: Oblivion (2006)

Outline

1. Techniques for UI design
2. Group activity: *discoverability*

Nielsen’s heuristics for UI design

1. Make system status visible
2. Match the real world
3. Provide control and freedom
4. Be consistent
5. Prevent errors when possible

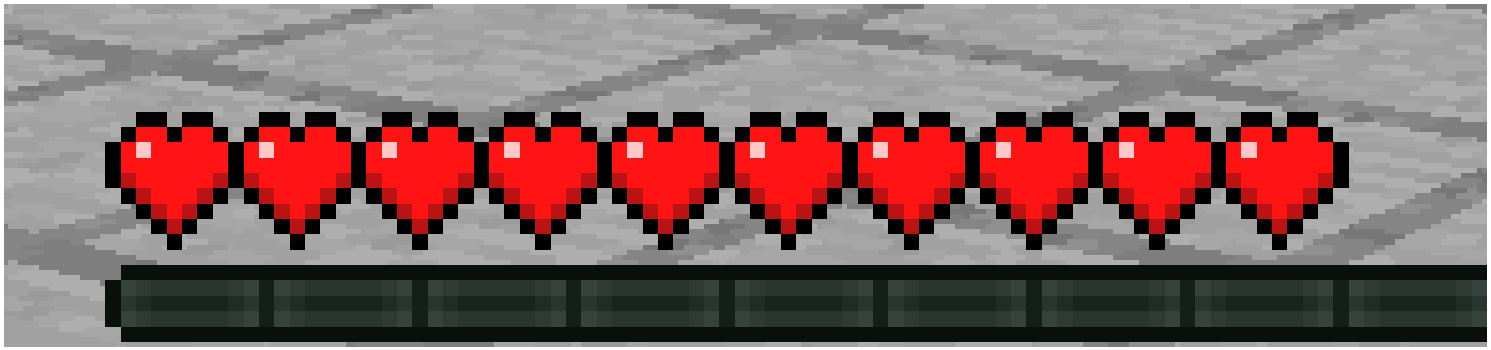
Nielsen’s heuristics for UI design

6. Facilitate recognition rather than recall
7. Be flexible and efficient
8. Use minimalist design
9. Help users recognize and recover from errors
10. Provide help and documentation

1. Make system status visible

The system should always **keep users informed** about what is going on, through **appropriate feedback** within reasonable time.

1. Make system status visible



Minecraft (2011)

1. Make system status visible

Half-Life (1998)

2. Match the real world

The system should speak the users' language, with words, phrases and **concepts familiar to the user**, rather than system-oriented terms. **Follow real-world conventions**, making information appear in a natural and logical order.

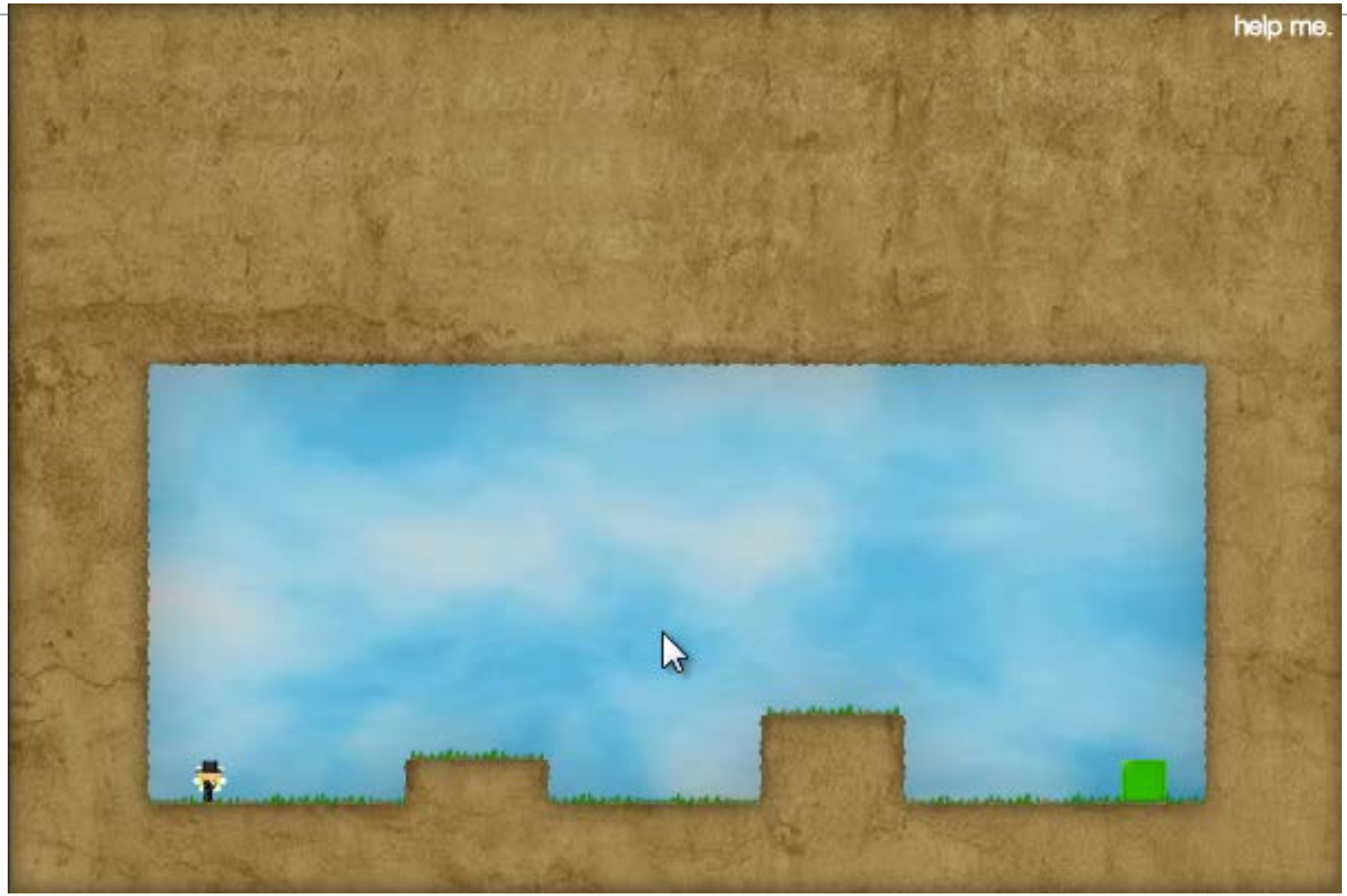
2. Match the real world

Braid (2008)

3. Provide control and freedom

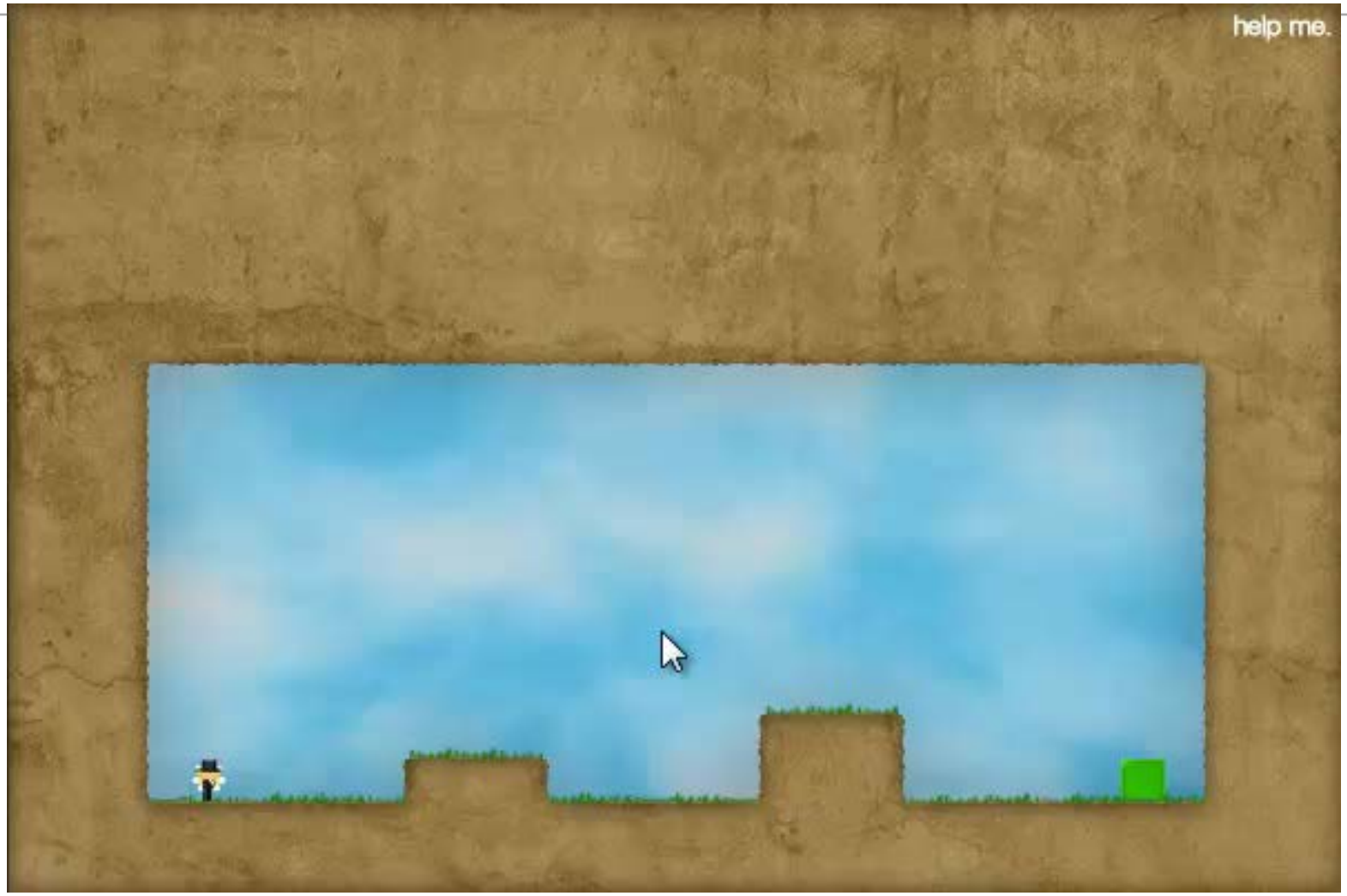
Users often choose system functions by mistake and will **need a clearly marked “emergency exit”** to leave the unwanted state without having to go through an extended dialogue. **Support undo** and redo.

3. Provide control and freedom



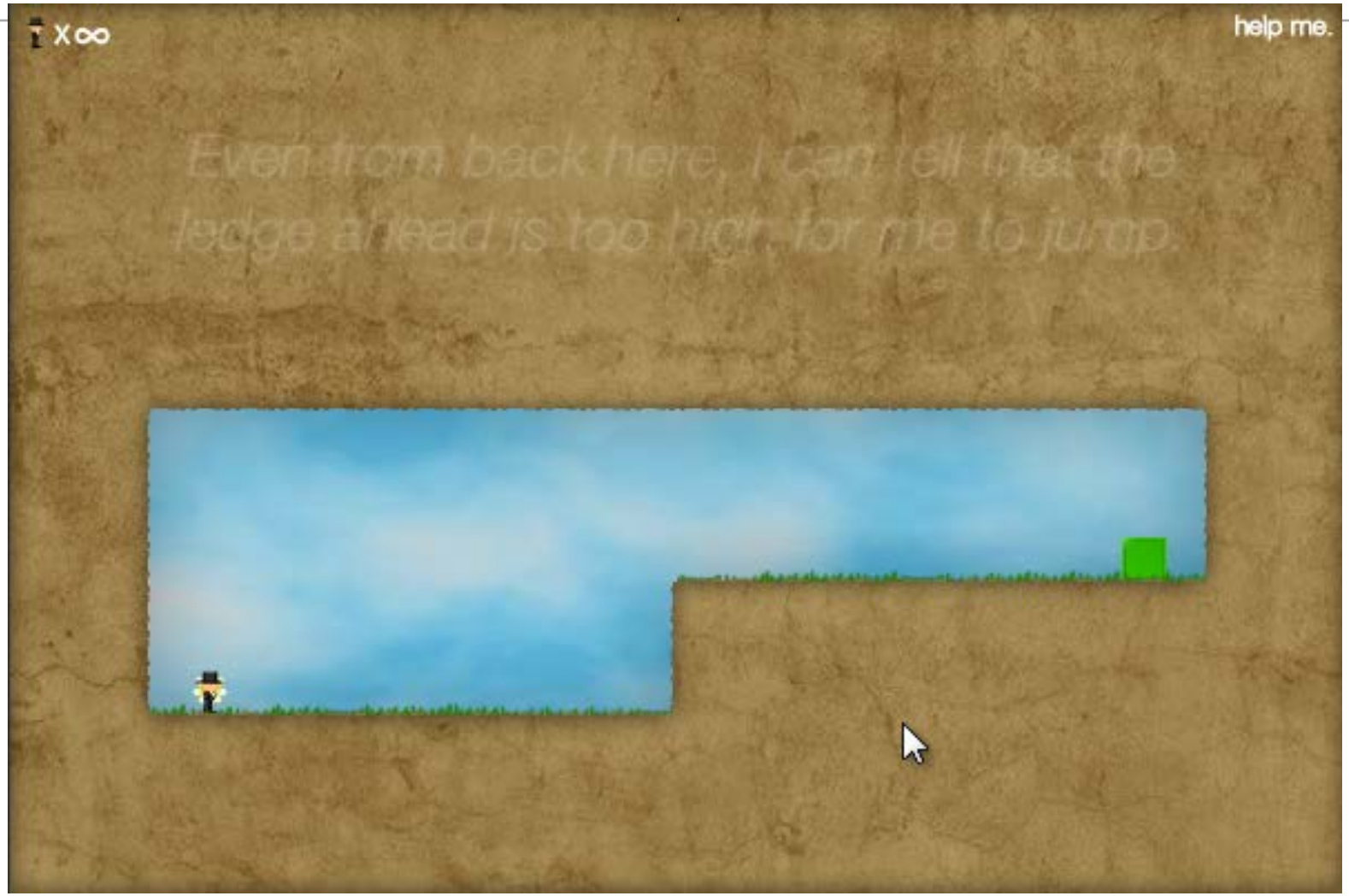
In The Company of Myself (2009)

3. Provide control and freedom



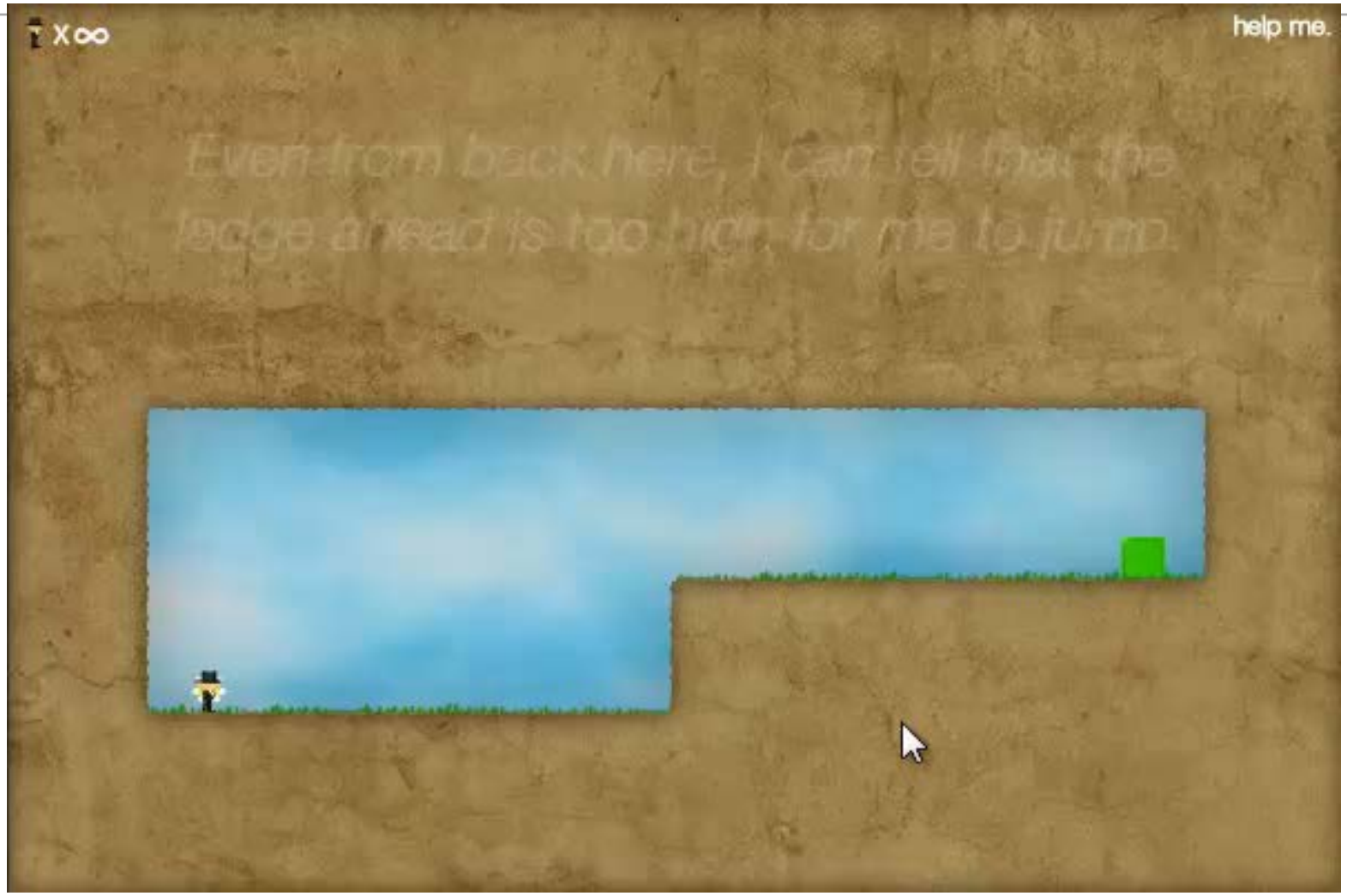
In The Company of Myself (2009)

3. Provide control and freedom



In The Company of Myself (2009)

3. Provide control and freedom



In The Company of Myself (2009)

Examples

- Make system status visible
- Match the real world
- Provide control and freedom

Evolution of Final Fantasy UI

Final Fantasy VII (1997)

Evolution of Final Fantasy UI

Final Fantasy VIII (1999)

Evolution of Final Fantasy UI

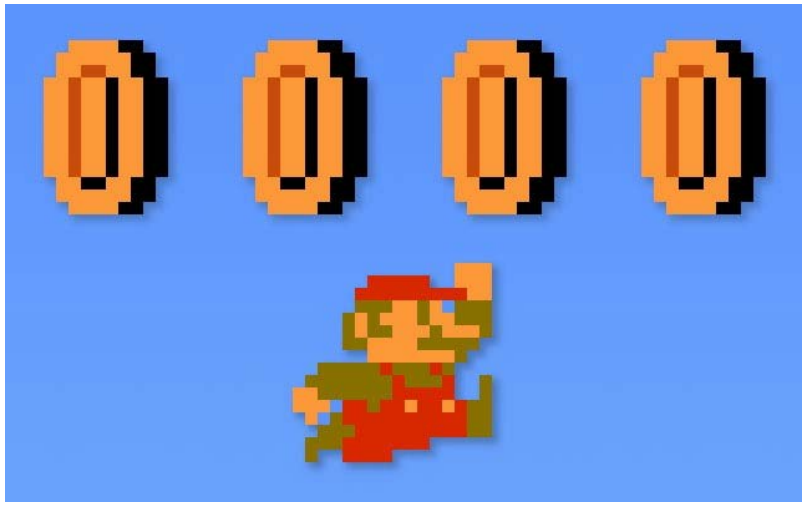
Final Fantasy XII (2006)

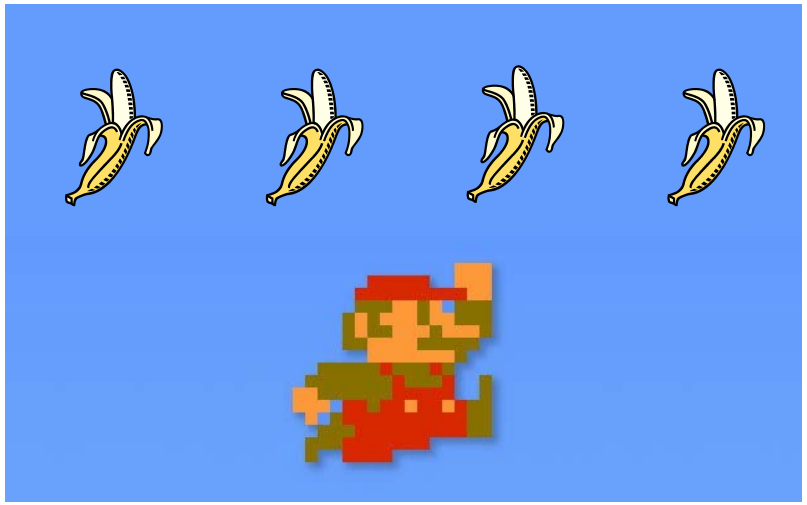
Evolution of Final Fantasy UI

Final Fantasy XIII (2009)

Examples

- Make system status visible
- Match the real world
- Provide control and freedom





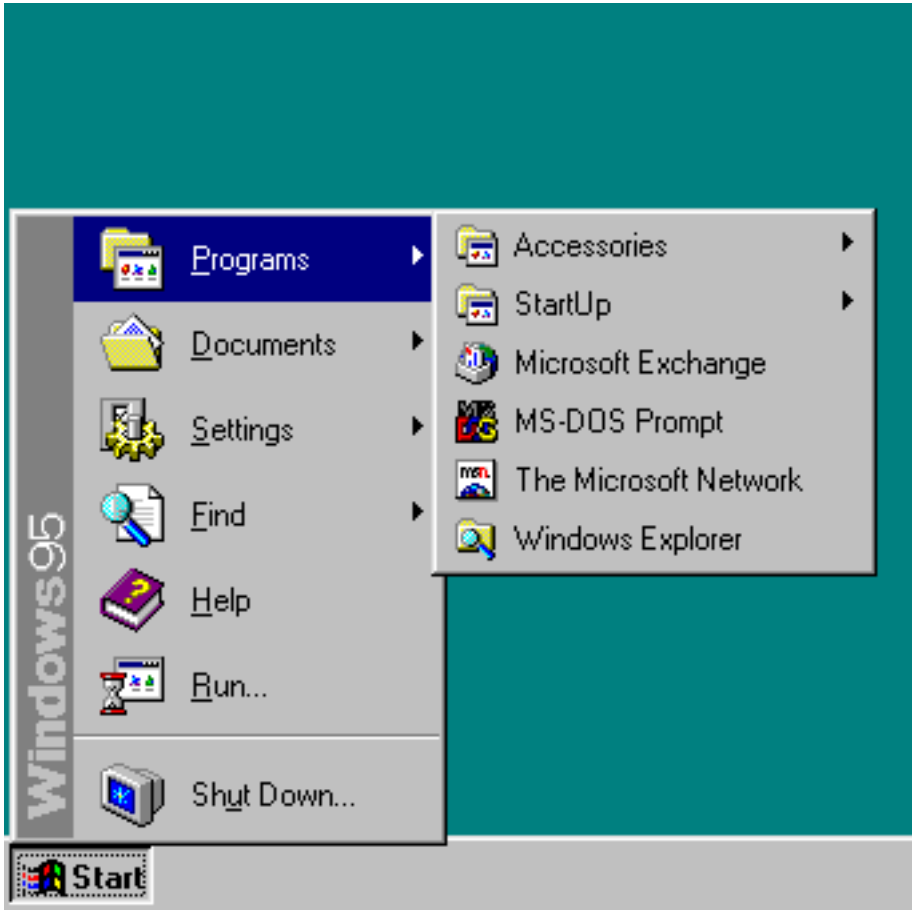
4. Be consistent

Users should not have to wonder whether different words, situations, or actions mean the same thing. **Follow platform conventions**.

5. Prevent errors

Even better than good error messages is a careful design which **prevents a problem from occurring in the first place**. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

5. Prevent errors



5. Prevent errors

6. Recognition rather than recall

Minimize the user's memory load by **making objects, actions, and options visible**. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

6. Use recognition rather than recall

6. Use recognition rather than recall

Plants vs. Zombies (2009)

Examples

- Make system status visible
- Match the real world
- Provide control and freedom
- Be consistent
- Prevent errors when possible
- Facilitate recognition rather than recall

King's Quest VI (1992)

Examples

- Make system status visible
- Match the real world
- Provide control and freedom
- Be consistent
- Prevent errors when possible
- Facilitate recognition rather than recall

Braid (2008)

7. Be flexible and efficient

Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. **Allow users to tailor frequent actions**.

8. Use minimalist design

Dialogues **should not contain information which is irrelevant or rarely needed**. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

8. Use minimalist design

9. Help users recover from errors

Error messages should be expressed in plain language, precisely **indicate the problem**, and constructively **suggest a solution**.

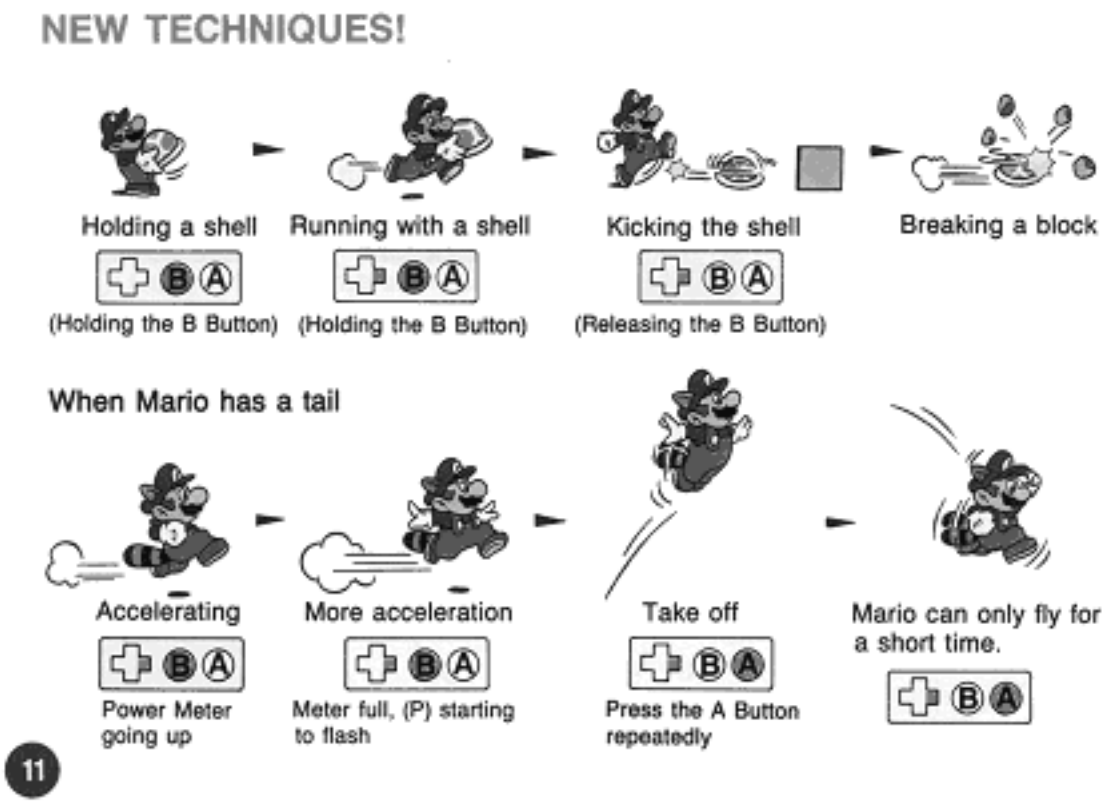
9. Help users recover from errors

Braid (2008)

10. Provide help and documentation

Even though it is better if the system can be used without documentation, it may be necessary to **provide help and documentation**. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

10. Provide help and documentation

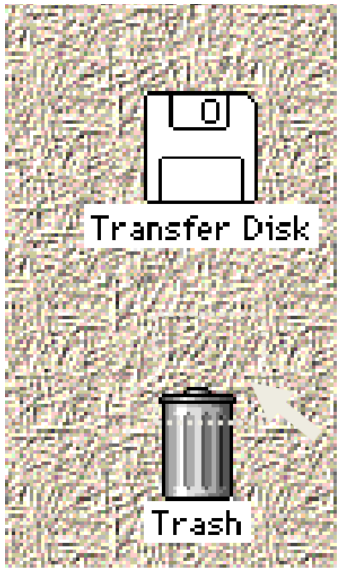


Examples

- Make system status visible
- Match the real world
- Provide control and freedom
- Be consistent
- Prevent errors when possible
- Facilitate recognition rather than recall
- Be flexible and efficient
- Use minimalist design
- Help users recognize and recover from errors
- Provide help and documentation

Examples

- Make system status visible
- Match the real world
- Provide control and freedom
- Be consistent
- Prevent errors when possible
- Facilitate recognition rather than recall
- Be flexible and efficient
- Use minimalist design
- Help users recognize and recover from errors
- Provide help and documentation



Nielsen's heuristics for UI design

1. Make system status visible
2. Match the real world
3. Provide control and freedom
4. Be consistent
5. Prevent errors when possible
6. Facilitate recognition rather than recall
7. Be flexible and efficient
8. Use minimalist design
9. Help users recognize and recover from errors
10. Provide help and documentation

Group Activity: Part 1

- Pick two *actions* in your game
  - Brainstorm tutorials for this
  - Sketch them (including UI)
- Pick two *interactions* in your game
  - Brainstorm *discoverable situations*
  - Sketch them (including UI)

Group Activity: Part 2

- Pick two *actions* in your game
  - Brainstorm tutorials for this
  - Sketch them (including UI)
- Pick two *interactions* in your game
  - Brainstorm *discoverable situations*
  - Sketch them (including UI)
- Show to a different group
- Use Nielsen's heuristics to evaluate