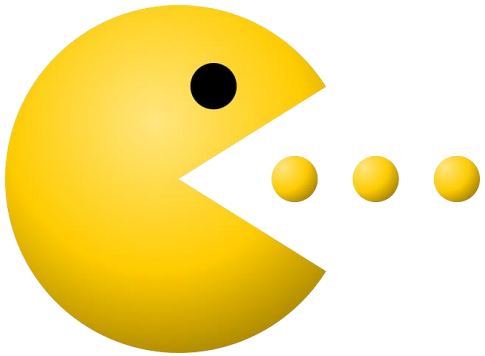
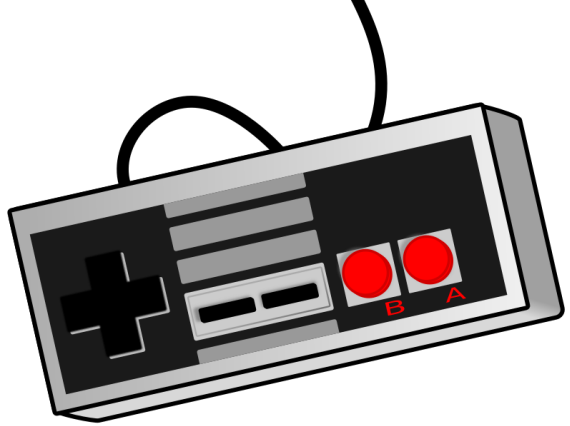




Lecture 03 -Level Design

PREPARED BY NICOLAS BERGERON

420-141-VA - GAME PROGRAMMING 1 - VANIER COLLEGE

Outline

Level Design

Variables and Methods

Level Design

Design Roles

- A level designer is a separate role from that of a game designer

Game Designer Categories

- System Designer
  - Defines the gameplay – the challenges the player will face and the actions taken to overcome these challenges.

- Level Designer
  - Creates the space in which the game takes place (its furnishings, backgrounds, much of its emotional context, etc.) as well as the player's moment to experience.

What Is Level Design?

- Level design: constructing the experience for the player using components from the game designer
- Level designers create:
  - Space in which the game takes place
  - Initial conditions of the level
  - Set of challenges within the level
  - Termination conditions of the level
  - Interplay between gameplay and story
  - Aesthetics and mood of the level

Key Design Principles

Universal level design principles

- Early levels of a game are tutorial levels
- Vary the pacing of the level
- When the player surmounts a challenge that consumes his resources, provide more resources
- Avoid things that don’t make sense
  - Don’t put rewards or dangers where a player would not reasonably expect to find them
- Clearly inform the player of his short-term goals

Key Design Principles (Cont.)

Universal level design principles (cont.)

- Be clear about risks, rewards, and consequences
  - Avoid “learn by dying” designs
- Reward the player for skill, imagination, intelligence, and dedication
  - Good players should be rewarded
  - E.g., power-ups/resources, shortcuts, secret levels, etc.
- Reward in a large way, punish in a small way
  - Give players an enjoyable experience; empathize with gameplay experience

Key Design Principles (Cont.)

Genre-specific level design principles

- Shooter game—reward precision and timing
- Action game—vary the pace
  - Allow player to rest between frenetic action
- Strategy game—reward planning
- Role-playing game—offer opportunities for character growth and player self-expression
- Sports game—correctness is vital
- Vehicle simulation—reward skillful maneuvering

Key Design Principles (Cont.)

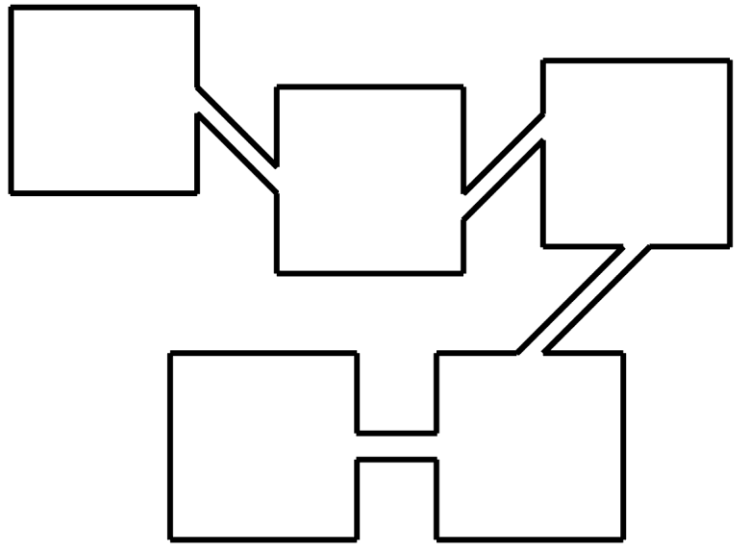
Genre-specific level design principles (cont.)

- Construction and management simulation—offer an interesting variety of initial conditions and goals
  - Chapter: start with partial construction to start from, with a goal (and perhaps time limit)
- Adventure game—construct challenges that harmonize with their locations and the story
- Artificial life game—create many interaction opportunities for the AI in their environment
- Puzzle game—give the player time to think
  - After a long time, offer hints (or let player buy hints)

Layouts

Linear layouts

- Require player to move in a fixed sequence
- Player can move only to next or previous area
- Used traditionally in side-scrolling action games and rail-shooters



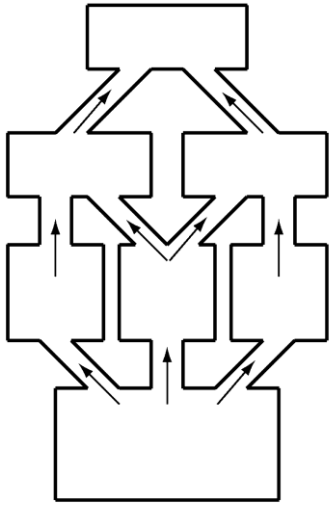
Open layouts

- Allow unconstrained movement
- Correspond to the outdoors
- Used in war games and role-playing games

Layouts (Cont.)

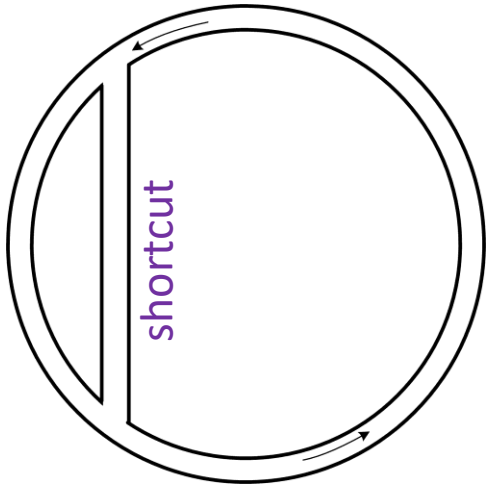
Parallel layouts

- Modern variant of linear layouts
- Variety of paths can go through the level
- Can reflect a foldback story structure



Ring layouts

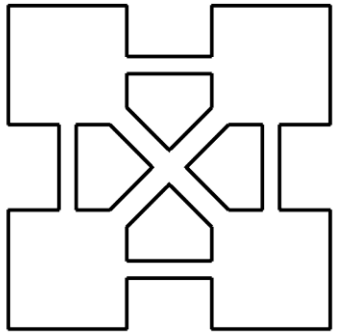
- Path returns to its starting point
- Oval tracks or twisting road-racing tracks are rings
- Used for racing games



Layouts (Cont.)

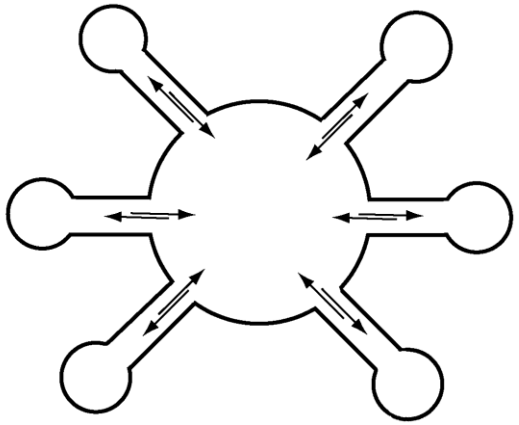
Network layouts

- Spaces connect to other spaces in different ways
- Give the player freedom to take any path
- Stories must be able to tolerate player experiencing events in any sequence



Hub-and-spoke layouts

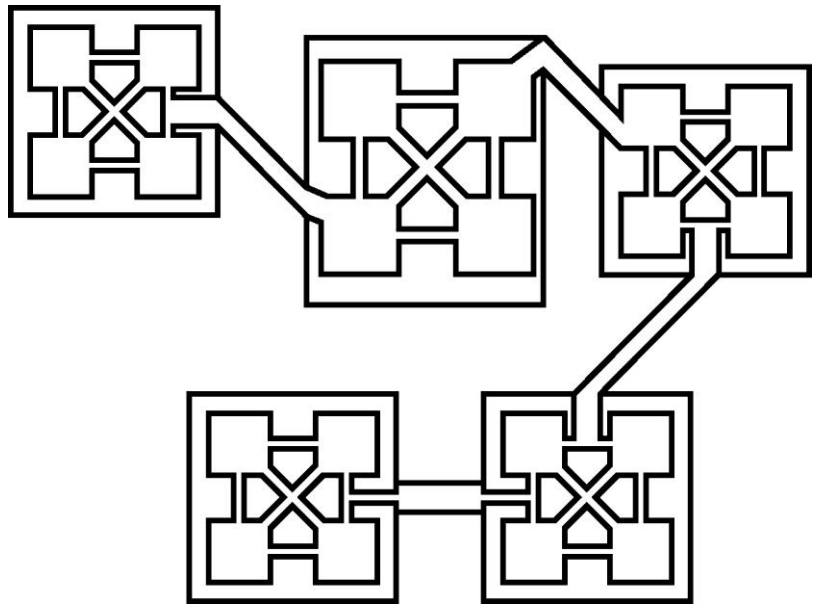
- Central hub is usually a safe zone
- Provides some choice of where to go
- Lock off some areas to control sequence a little



Layouts (Cont.)

Combinations of layouts

- Combines aspects of several layout types
- Role-playing games and adventure games often use combination layouts



Expanding on the Principles

Level designer assembles components to create the atmosphere

- Lighting
- Color palette
- Weather and atmospheric effects
- Special visual effects
- Music
- Ambient audio
- Special audio effects

Expanding on the Principles (Cont.)

In designing progression, consider these factors:

- Mechanics
- Experience duration
- Ancillary rewards and environmental progression
- Practical gameplay rewards
- Difficulty (more on this next class)
- Actions available to the player
- Story progression
- Character growth

Expanding on the Principles (Cont.)

Pacing refers to the frequency of individual challenges

- Genre affects pacing
  - Multiplayer deathmatch shooters use the fastest pace
  - Adventure games use the slowest pace
- Vary the pacing with fast and slow periods
  - Slow periods to recover and find resources
- Overall pacing should remain steady or become more rapid near the end of the level
  - Commonly a boss to defeat at end of level

Expanding on the Principles (Cont.)

Tutorial levels teach the player how to play

- Somewhat scripted experience that explains the game's user interface, challenges, and actions
  - Introduce features in an orderly sequence
  - Switch off features not yet introduced
  - If the game is complex, use more than one tutorial level
  - Highlight screen elements when you introduce them
  - Let player go back and try things again conveniently
- Voiceover narration or text can explain the game

Design rule: Make tutorial levels *optional*

Extra Credits on Tutorial Level Design

https://www.youtube.com/watch?v=ZH2wGpEZVgE

The Level Design Process

Design to level design handoff

- List features in the level
- Create a rough overview map of the level

Planning phase

- Plan the level in detail
- Plan the sequence of events
- Document the gameplay, art, performance, and code requirements

The Level Design Process (Cont.)

Prototyping (Grey Boxing)

- Construct temporary models of the landscape and objects
- Models serve as blueprints for the art team

Level review

- Review the prototype
- Feedback from the design, art, programming, audio, and testing teams

The Level Design Process (Cont.)

Level refinement and lock-down

- Make corrections based on feedback
- Review and correct until the level is flawless
- Lock the level

Level design to art handoff

- Give all files to the artists
- Tell artists how the level should look and work
- Contact the audio team for any needed audio

The Level Design Process (Cont.)

First art and rigging pass

- Art team builds the real artwork
- Technical Artists incorporate the new content into Game Engine

Art to level design handoff and review

- Receive final artwork
- Conduct a review

The Level Design Process (Cont.)

Content integration

- Assemble all the assets into the completed level
- Adjust any rigging as necessary

Bug fixing

- Test for bugs and mistakes and hand off to Quality Assurance (QA)

User testing and tuning

- QA creates a test plan and begins testing (more on testing next week)

Pitfalls of Level Design

Get the scope right

- Design within your available resources
- It’s common to design something that’s too big

Avoid conceptual non sequiturs

- Illogical events in a game make it harder to play
- Players should be rewarded, not punished, for using their intelligence

Pitfalls of Level Design (Cont.)

Don’t show the player everything at once

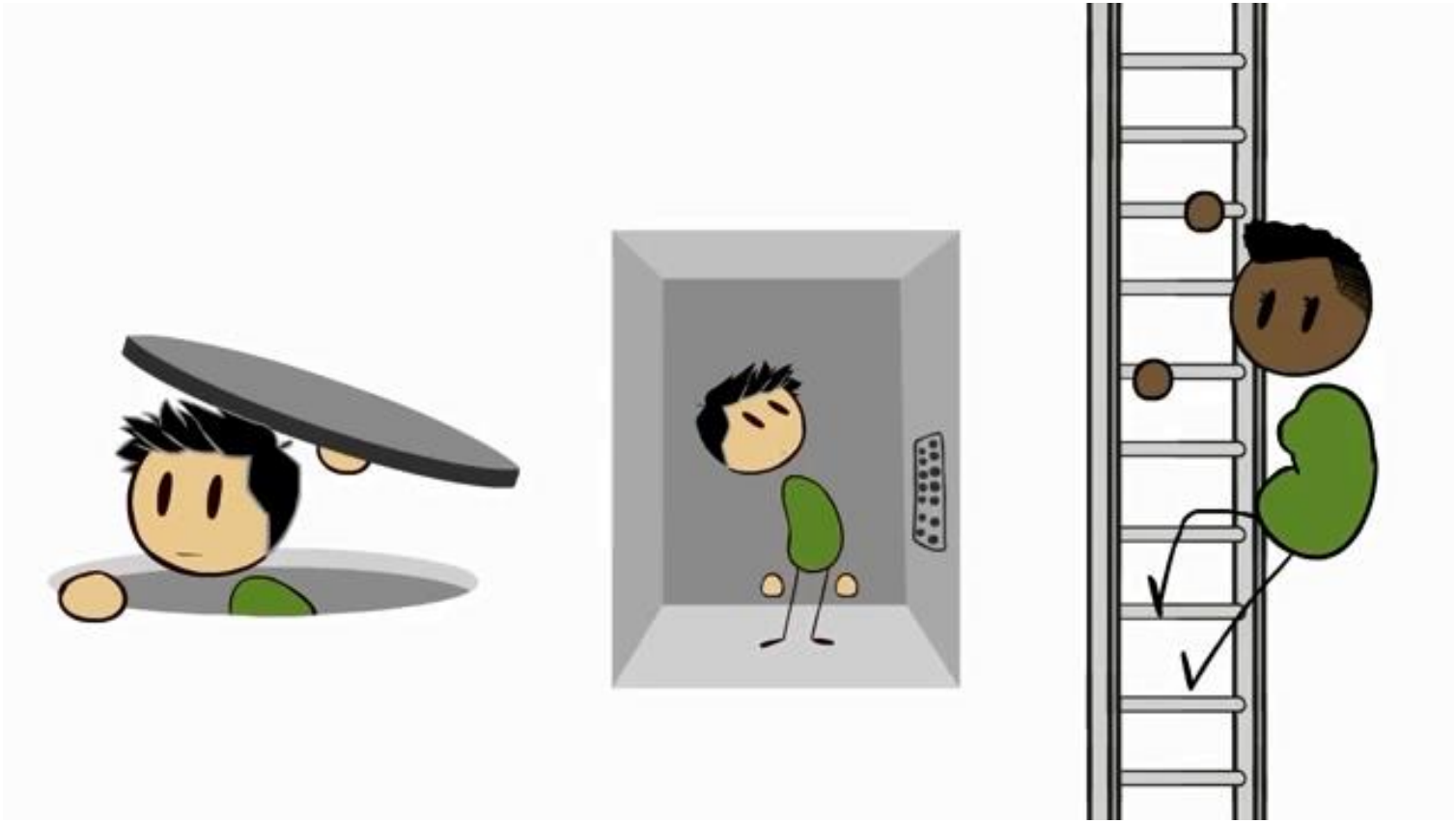
- Introduce new features gradually
- Maintain the player’s interest

Never lose sight of your audience

- Player-centric approach
- Understand your target audience and deliver what they want

Read the “Bad Game Designer, No Twinkie” articles for more examples of design errors

Extra Credits on Level Design



https://www.youtube.com/watch?v=-H97gCCJFXA

Java and Stride:
Variable types and methods

Primitive vs Object Types

Primitive type variables (boolean, int, double, …)

- Start with Lowercase character (convention)
- Values are encoded directly in the variable

Object types

- Start with Uppercase character (convention)
- Values are referenced, variables encodes the memory location of the objects, **not** the object values directly
- Object variables may have no value, also called the **null** reference

We can inspect objects in Greenfoot, notice the difference between primitive types and objects

Methods

Writing new methods allow creating new behaviors for objects

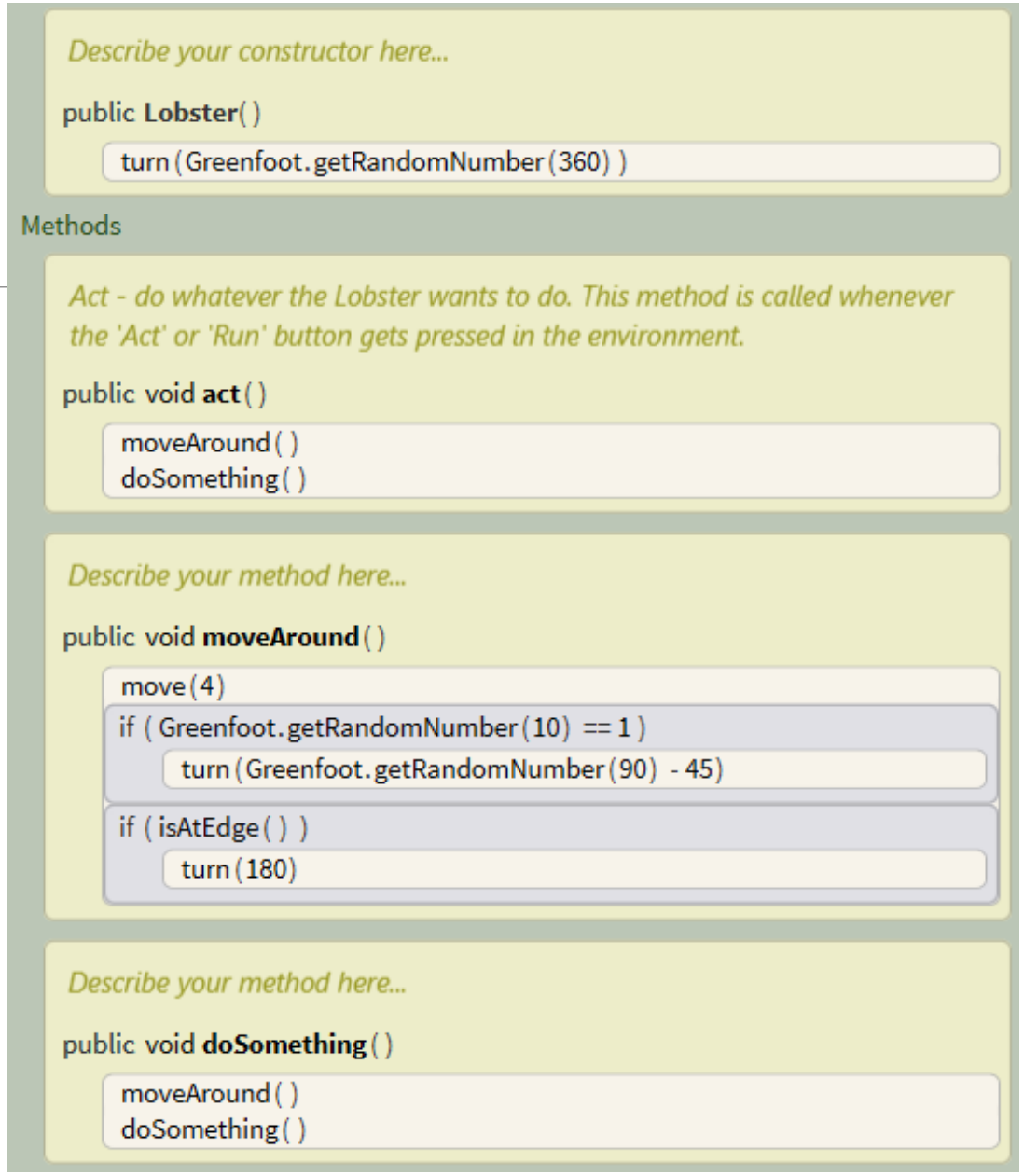
Writing new methods alone will not change the code being executed

Methods must be invoked (or called) in order to be executed

In Greenfoot, Actors and Worlds will have the act() method invoked every time step. This is a good place to invoke your methods!

Exercise: Why is the code on the right crashing?

```
at Lobster.doSomething(Lobster.java:49)
at Lobster.doSomething(Lobster.java:49)
at Lobster.doSomething(Lobster.java:49)
```



Questions

?