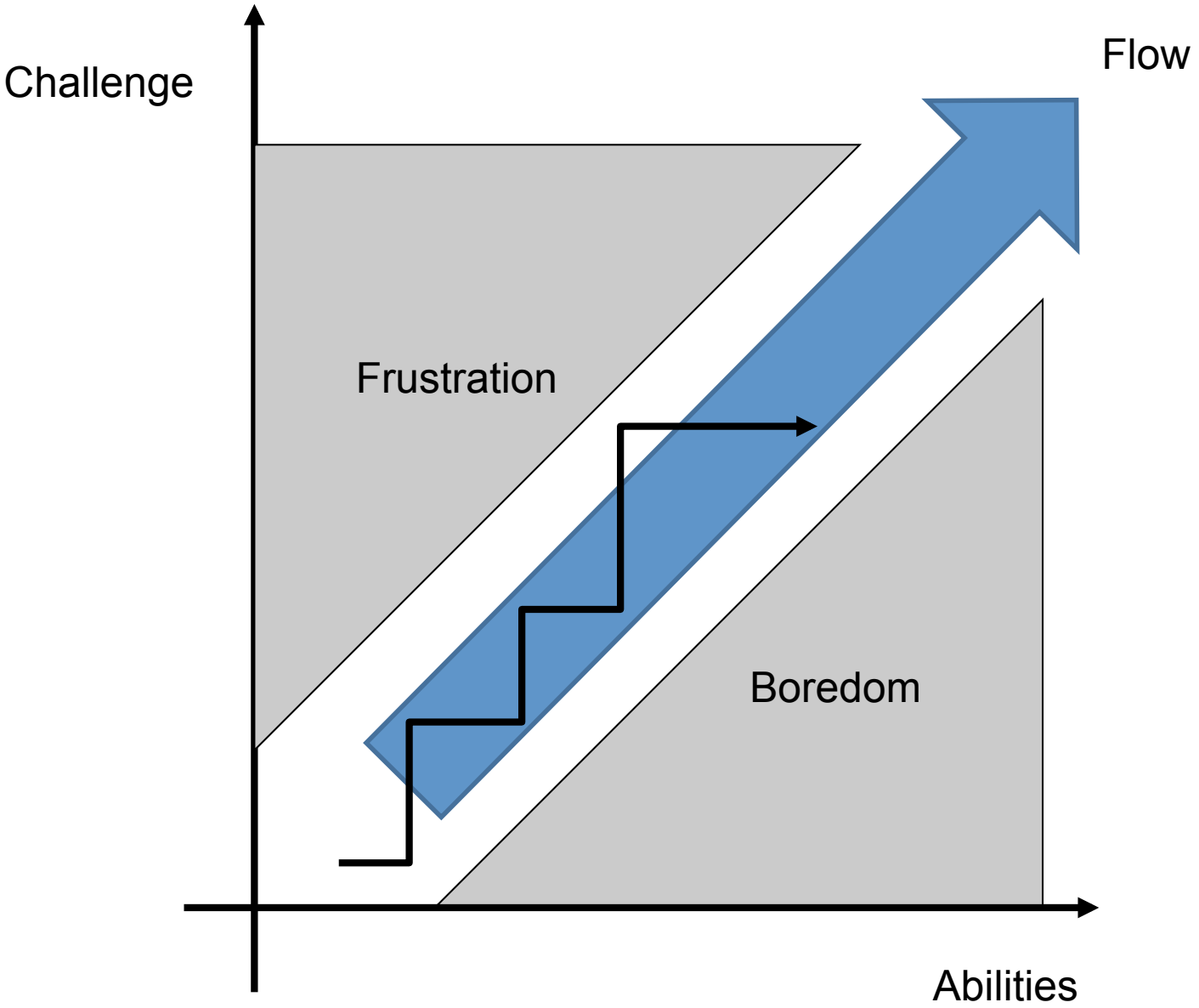
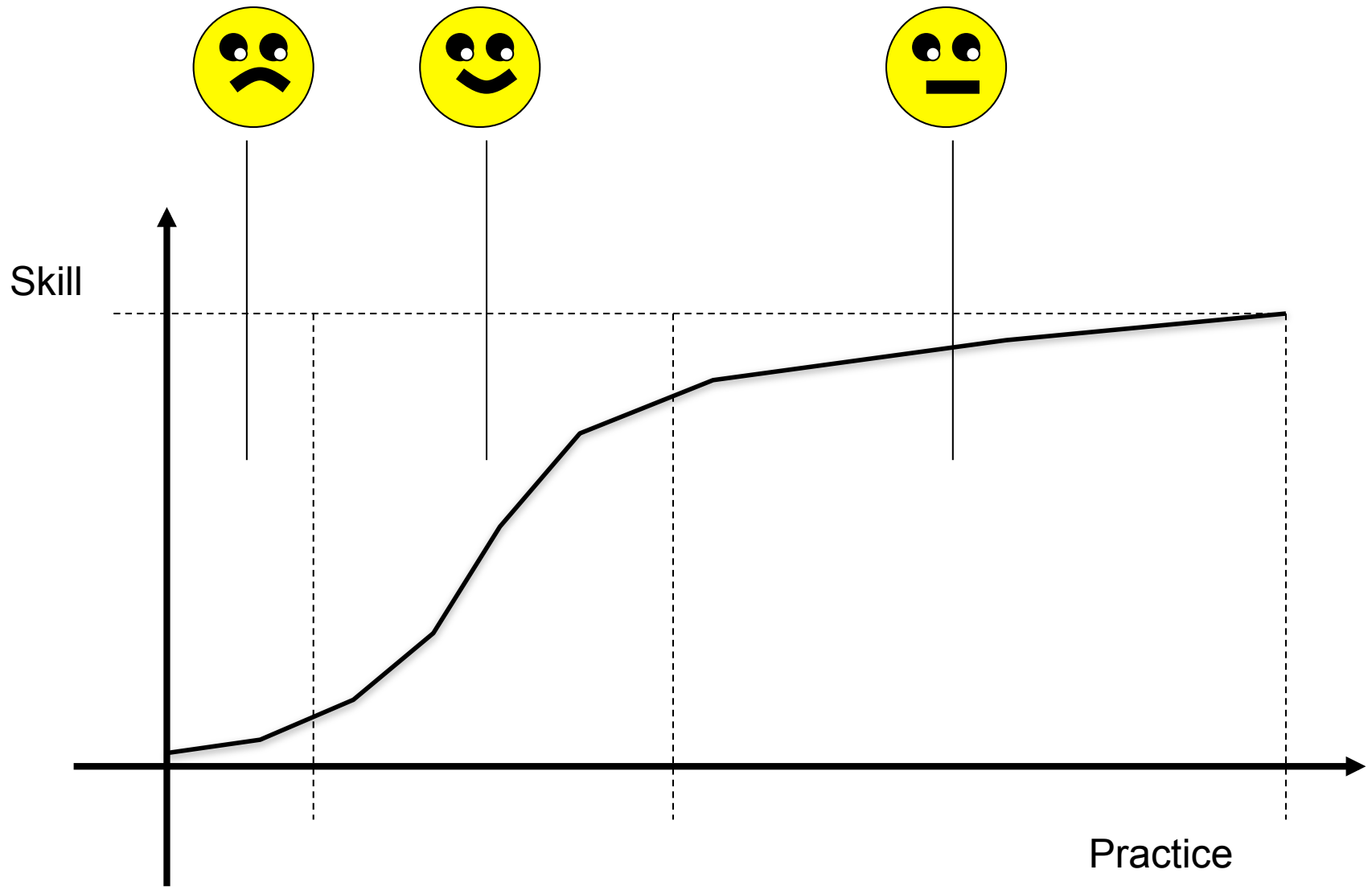
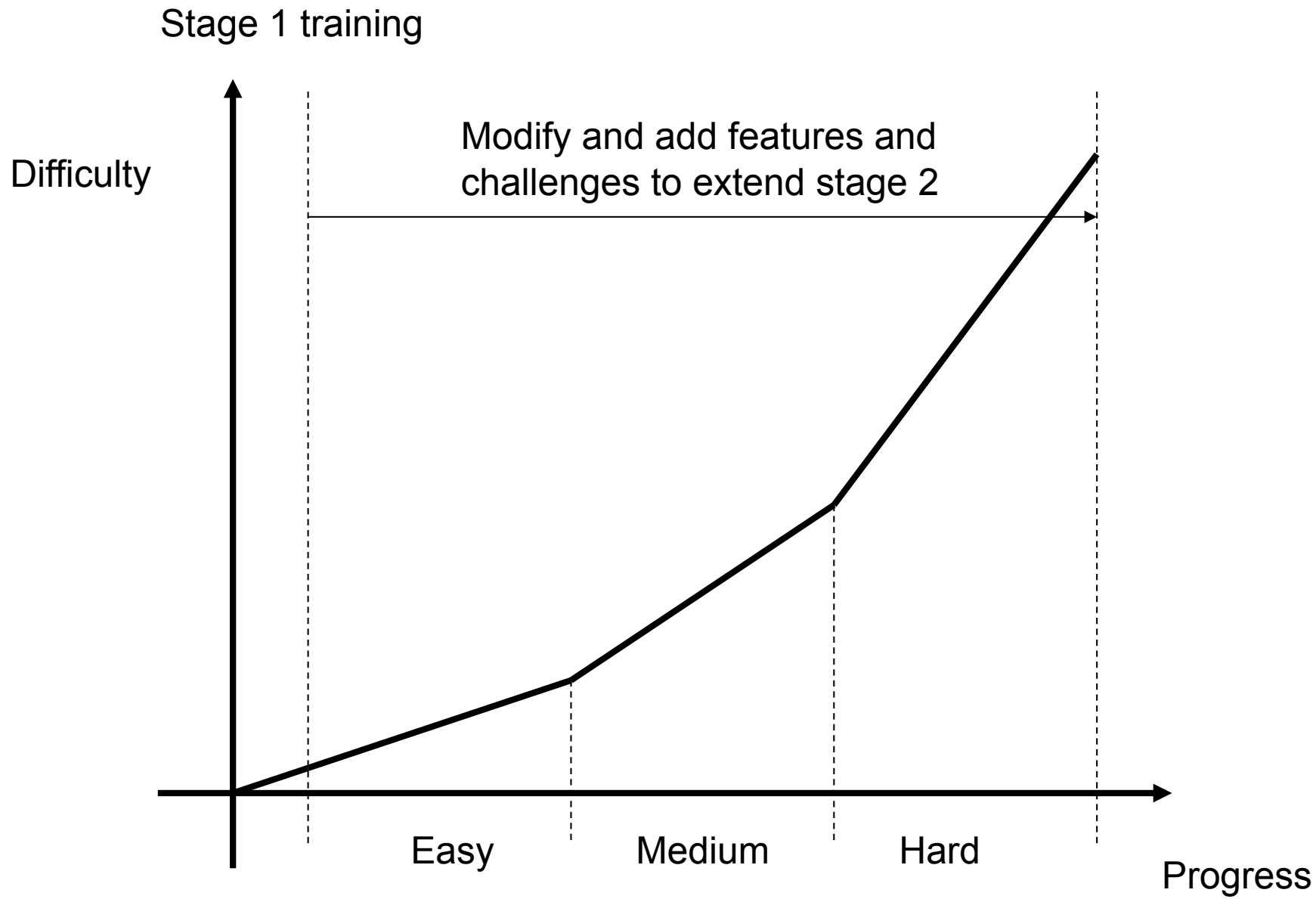
G54GAM - Games

- Balance
- Software architecture


Challenge
Flow
Frustration
Boredom
Abilities


Skill
Practice


Stage 1 training
Difficulty
Modify and add features and
challenges to extend stage 2
Easy
Medium
Hard
Progress

This week

- Balancing a game
  - Why is it important?
  - Understanding balance
    - Pay-off matrices
    - Dominant Strategies
    - Static and dynamic balance
  - Balancing Techniques
- System architecture
  - Structure and the Game Loop

Balance

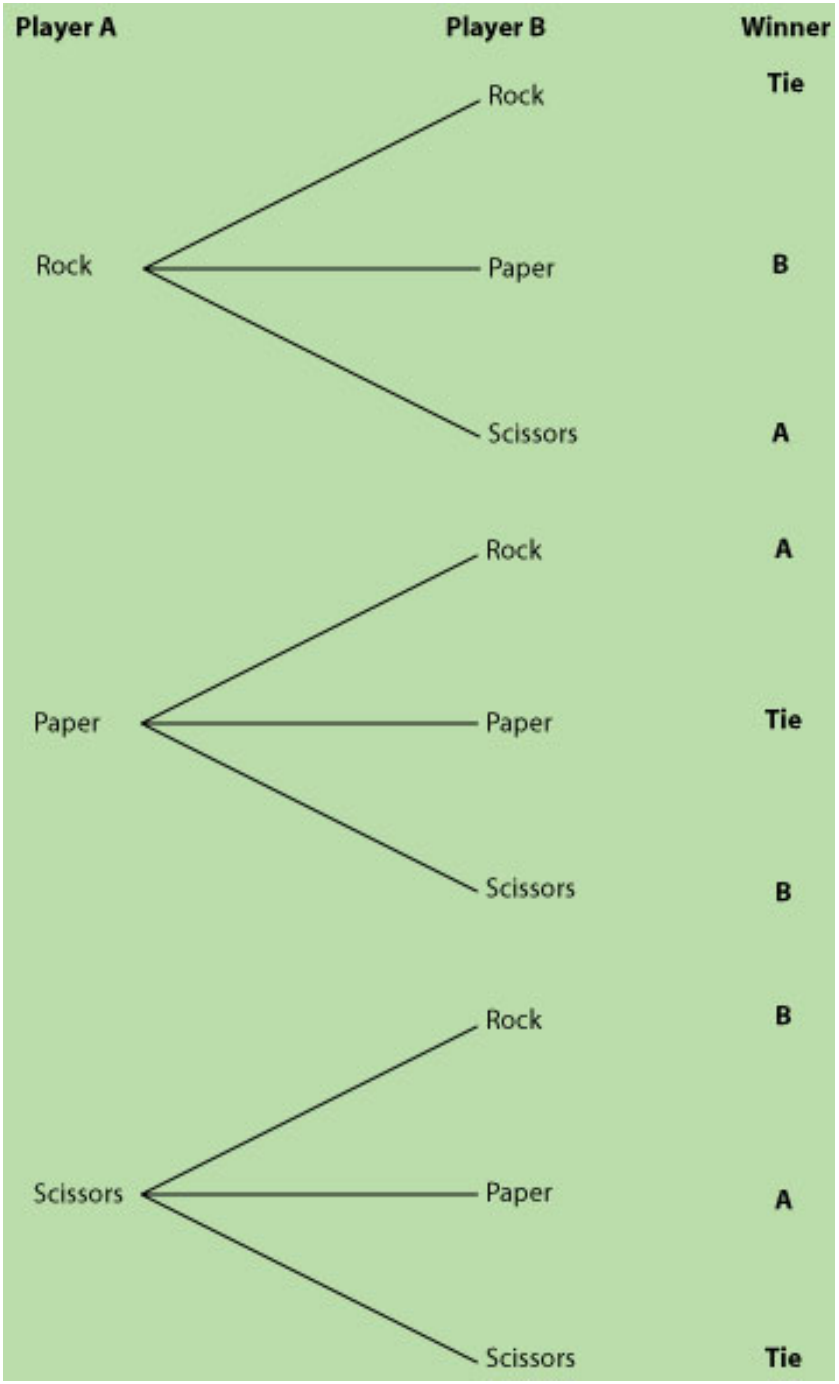
- Can make or break a game
  - Look, sound and even play well
  - Can still be a failure
- We may have all the formal and dramatic elements of game play
  - Need to be in balance with one another and the player
  - Game fails if they are not, no fun
- A **balanced** game is one where success of the player is largely determined by the skill of the player
  - Random events may occur
  - In general a better player should get further than a poor player

Is it balanced?

- Combinatorial game theory
  - Analyse
  - Optimisation problem
  - Just because a result is mathematically correct does not mean it is aesthetically pleasing
- Trial and error
  - Play, tweak, play, tweak…
  - Run out of time, release game
  - Tweak further by releasing additional patches
- Need to understand what we're balancing and how

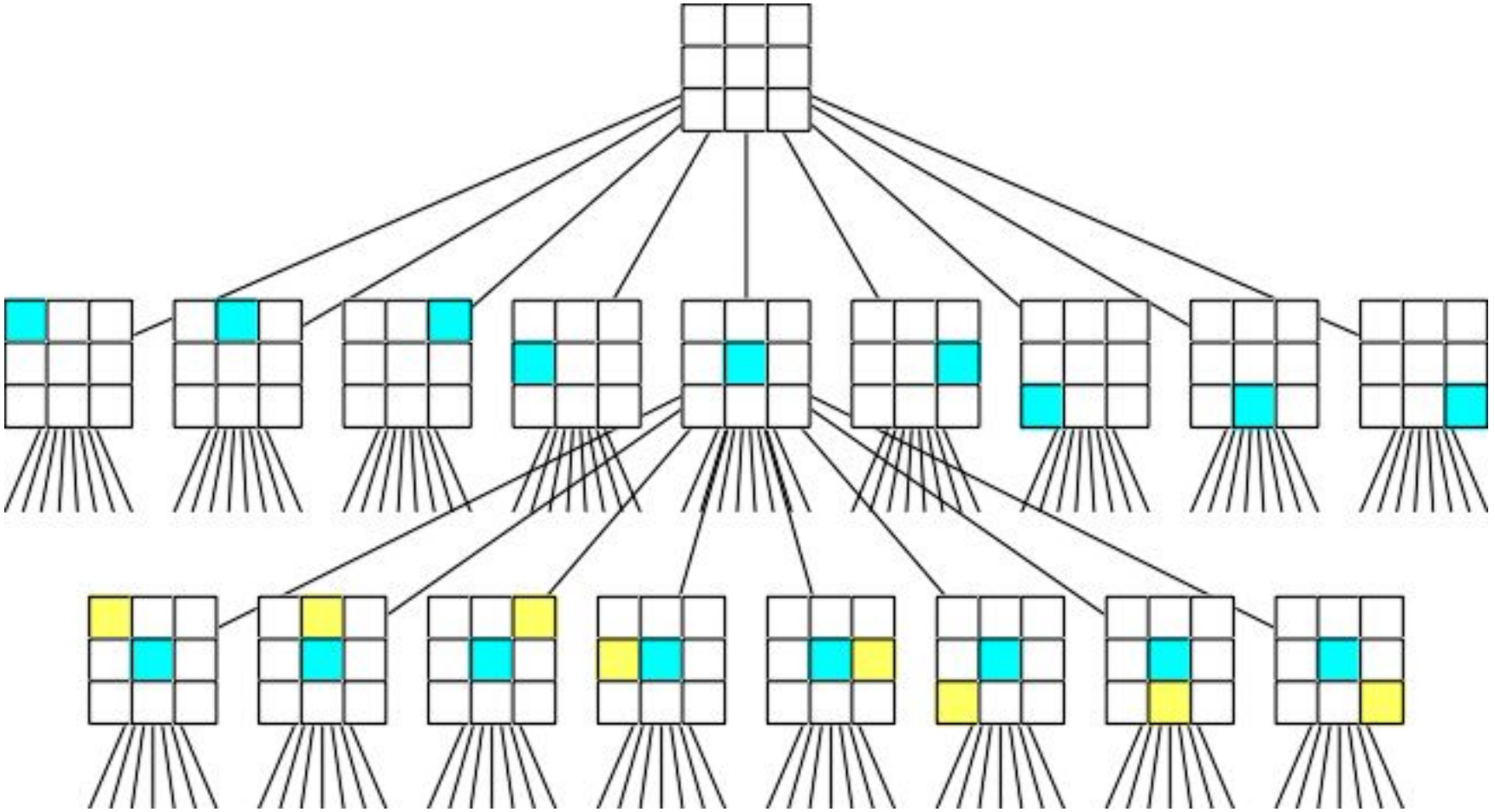
Is it balanced?

- **Static** balance
  - Are the rules fair when considered as a static system?
  - Is the initial state of the system (formal) balanced?
- **Dynamic** balance
  - Is an equilibrium maintained?
  - How does balance change with time and player interaction?
- (remember mechanics, dynamics, aesthetics)


Player A
Player B
Winner
Rock
Rock
Tie
Paper
B
Scissors
A
Paper
Rock
A
Paper
Tie
Scissors
B
Scissors
Rock
B
Paper
A
Scissors
Tie

Rock, paper, scissors – is it balanced?

	Scissors	Paper	Rock
Scissors	0	1	-1
Paper	-1	0	1
Rock	1	-1	0



Birthday Conundrum

- If it is my birthday, and you buy me flowers, you win 10 brownie points, because you remembered my birthday.
- If it's not my birthday, you will win 20 brownie points, because you have surprised me with your thoughtfulness.

Birthday Conundrum

	Birthday	Not Birthday
Buy Flowers	10	20
Do not buy flowers	-100	0

Dominant Strategies

- Always buy flowers
  - Always get positive payoff
- Never buy flowers
  - Zero payoff
  - Massive loss
- **Strongly** dominant strategy
  - Guarantees winning every time
- **Weakly** dominant strategy
  - Guarantees not losing, but drawing – Tic-tac-toe!
- All other strategies **recessive**
  - Why would a player choose to do something else?

Warcraft – always bet on the Orc

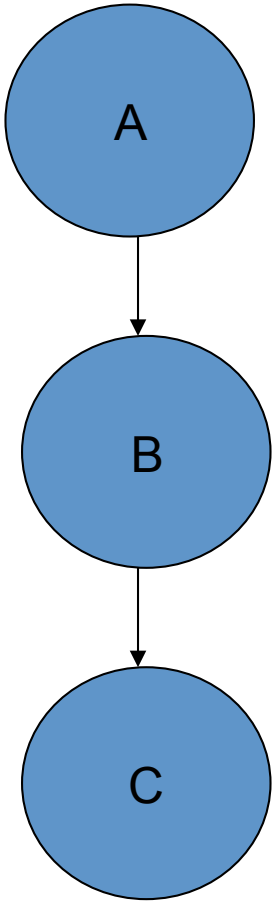

Attacking
25048
Selected across screen.
Apocalypse Tank

Balancing Techniques - Symmetry

- Each player (including the computer) is given the same starting conditions and abilities
- Most applicable to…
  - Sports simulations
  - Multi-player games
- Difficult to achieve precisely
- Leads to boring game play?

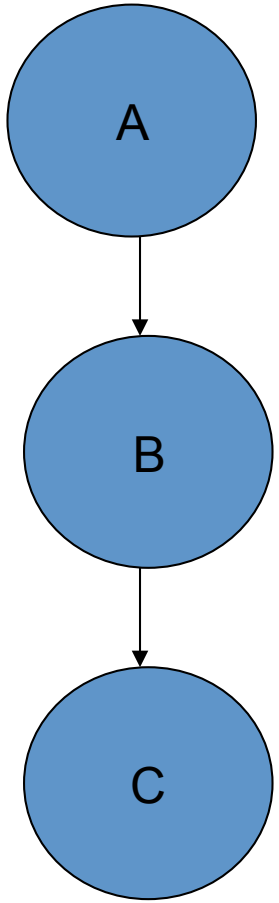
Transitive Relationships

- A one-way relationship between objects
- A beats B, B beats C, C beats nothing at all
- Why would anyone want C?



Transitive Relationships

- Make C **free**, and A **cost** something
- Reward without cost leads to a **dominant strategy**
- TRs continually drive a player towards a goal
  - Progression + regression
  - Any game that involves upgrading or augmenting player abilities

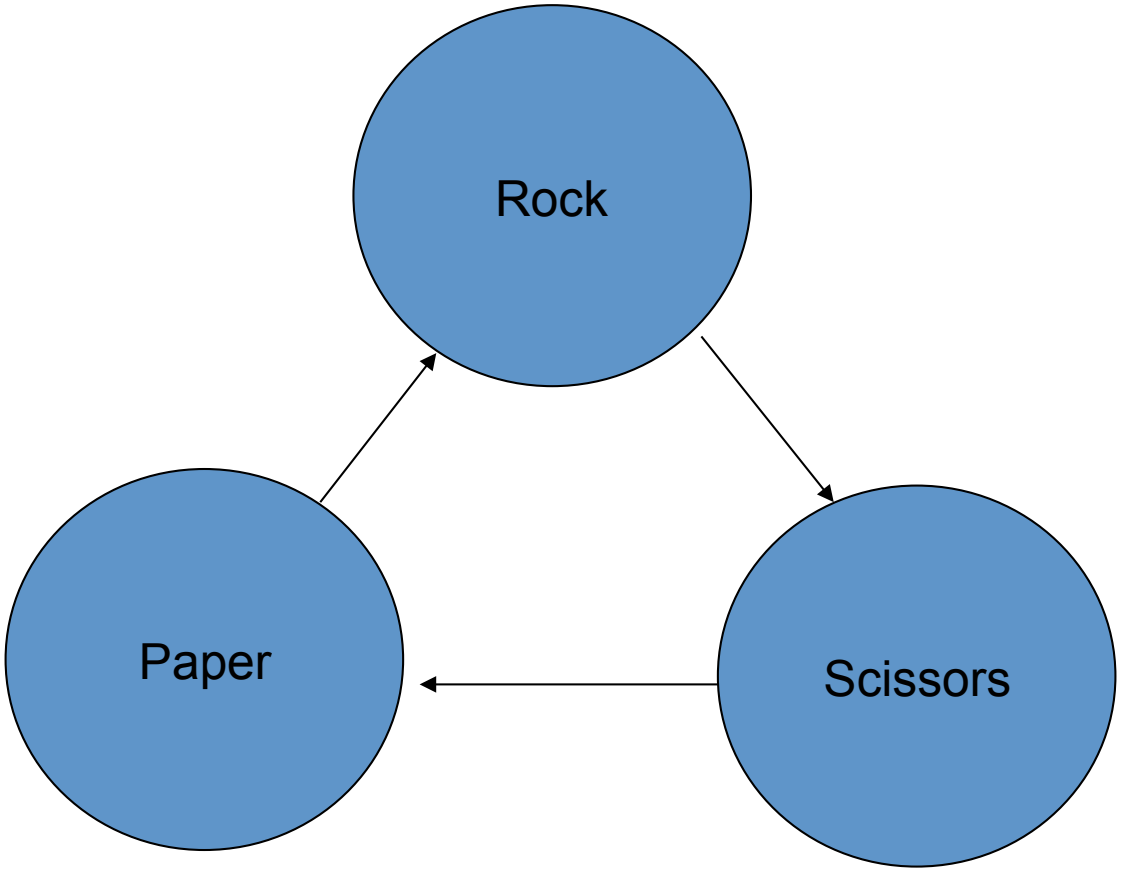


Transitive Relationships

	A	B	C
A	0	1	1
B	-1	0	1
C	-1	0	0


Gamer's Hell

Intransitive Relationships



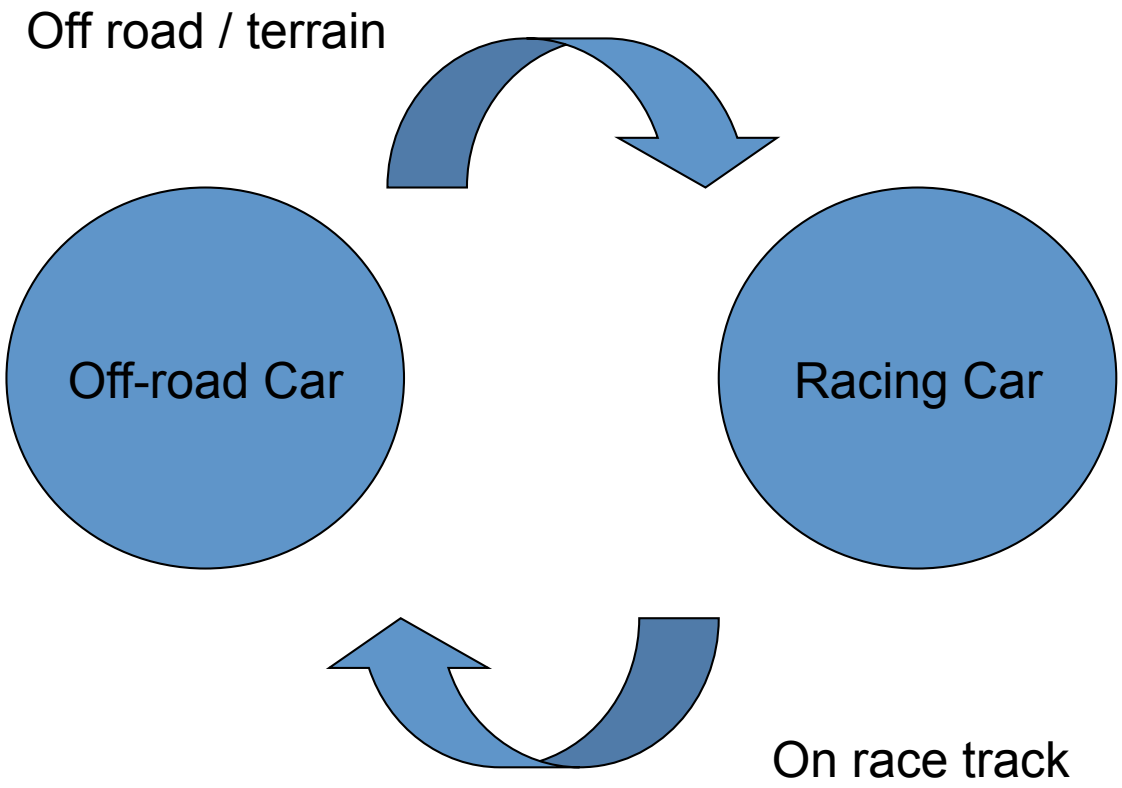
Intransitive Relationships

	Scissors	Paper	Rock
Scissors	0	1	-1
Paper	-1	0	1
Rock	1	-1	0

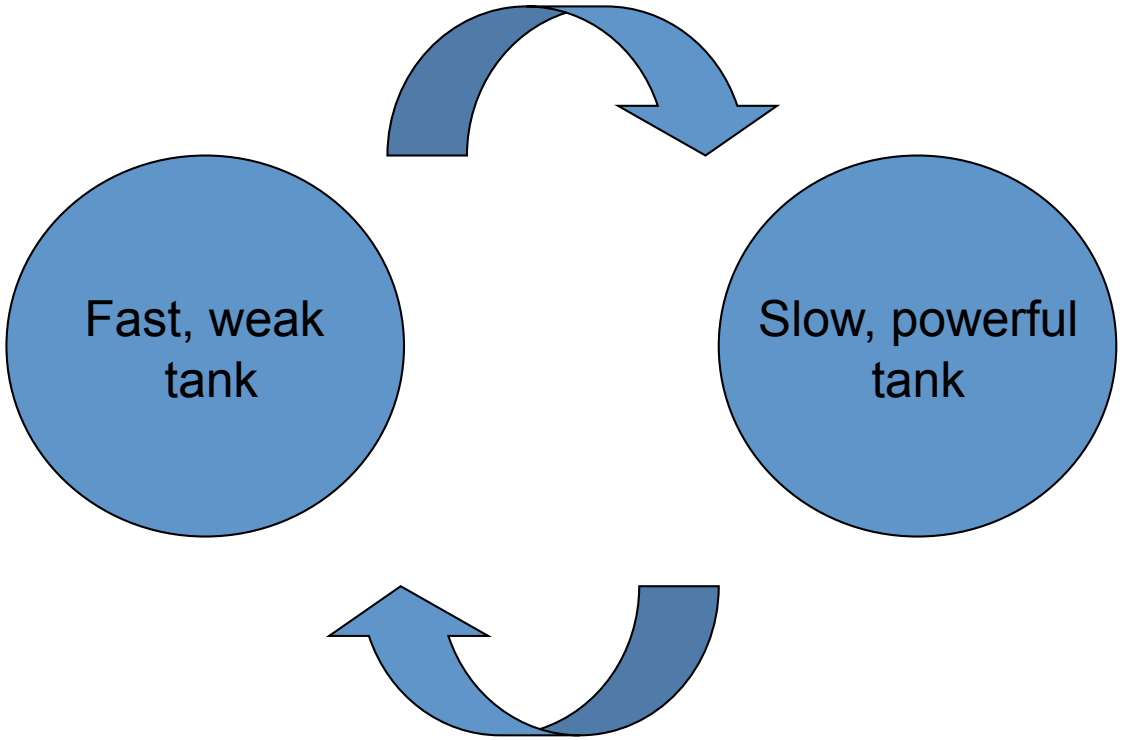
Intransitive Relationships

- Aesthetically pleasing
  - The game “looks” balanced
- Players quickly learn to understand the relationships
  - Does not lead to innovative game play
- Challenge the player to consider different environments

Intransitive Relationships



Trade-Offs



Intransitive trade offs

- Common in role playing games
  - "Trade off" one ability against each other
    - The player must decide which ability to maximise
  - Spend points on strength or charisma?
    - Stats-jugglings
    - Skills are independent and orthogonal
  - Still needs to be balanced
    - A strength point should given an equivalent advantage as a charisma point
    - Must still be able to complete the challenge
      - Arbitary punishment for making the wrong decision

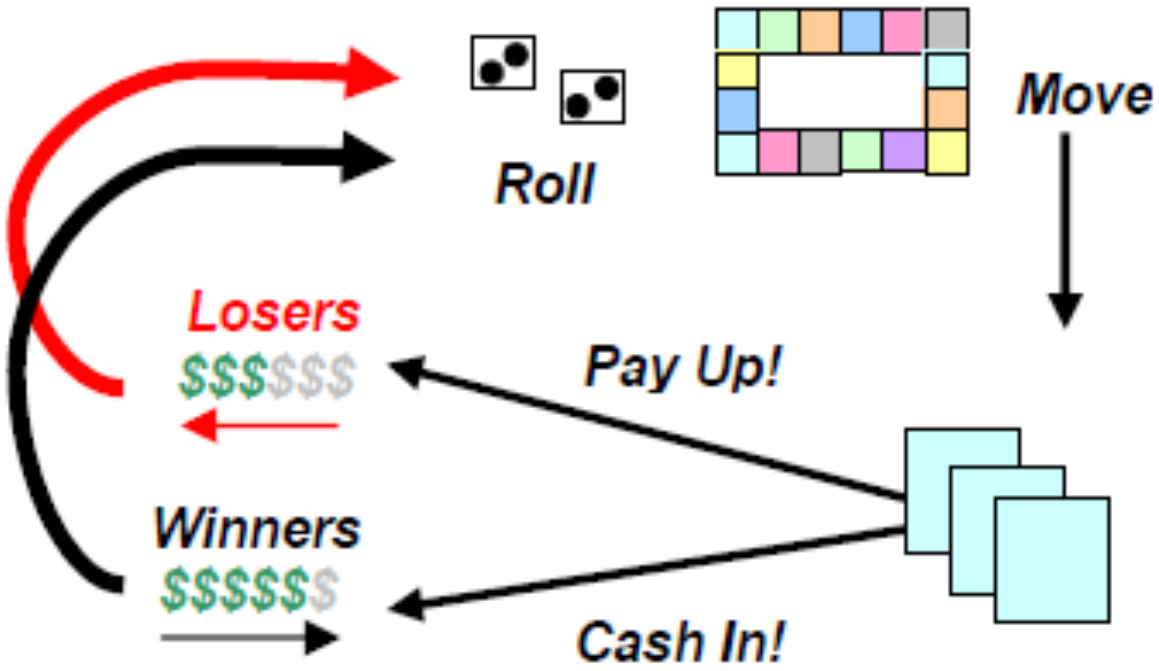
Dynamic Balance

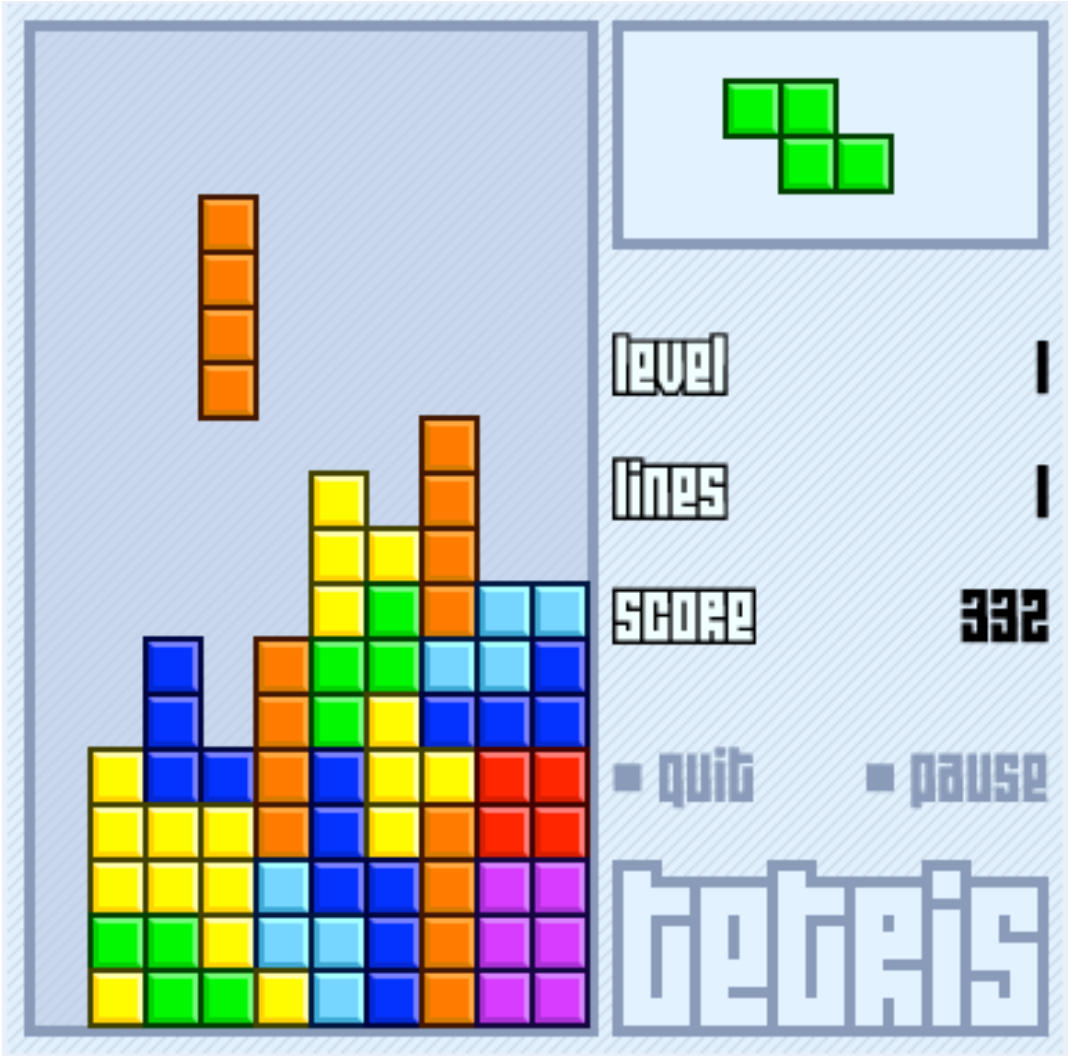
- As **time** and **player interaction** continue, what happens to the balance?
  - Is it maintained?
  - Is it destroyed?
  - How is it restored?
- How the game is dynamically balanced defines the game play of the game
  - Balance is disrupted – the player wins
  - Balance is maintained – the player can continue to play

Feedback

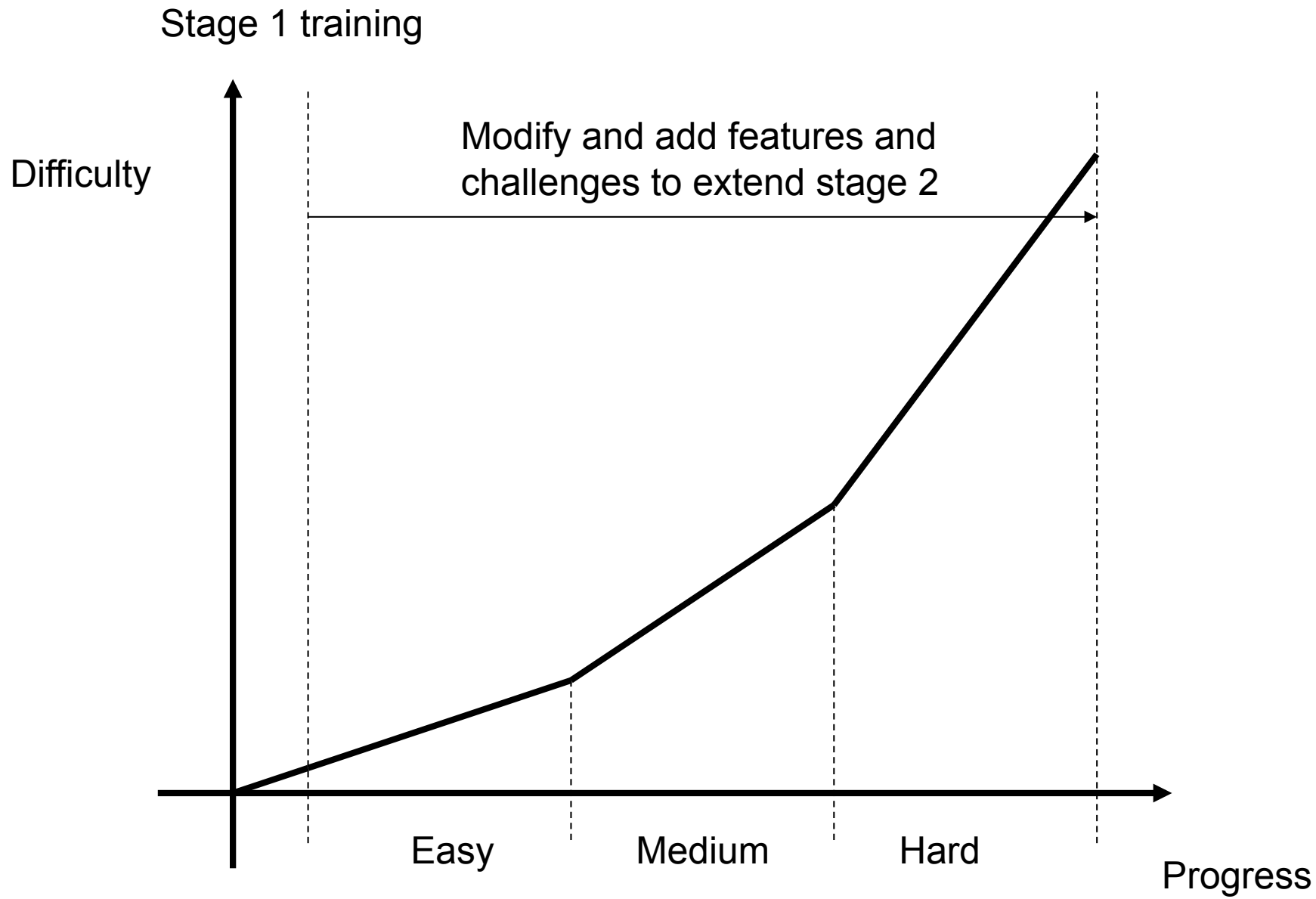
- Positive Feedback
  - Destabilises the game
  - Rewards the winner
  - Ends the game
  - Magnifies early successes
- Negative Feedback
  - Stabilises the game
  - Forgives the loser
  - Prolongs the game
  - Magnifies late successes
- Explicit user interaction

Predict and Describe Dynamics




level
1
lines
1
score
332
quit
pause
tetris


LAP
1
4
BG-345
+3.8
74
POS
7
8
BEST
0.00.00
0.27.7
448 KM/H
98%


Stage 1 training
Difficulty
Modify and add features and
challenges to extend stage 2
Easy
Medium
Hard
Progress

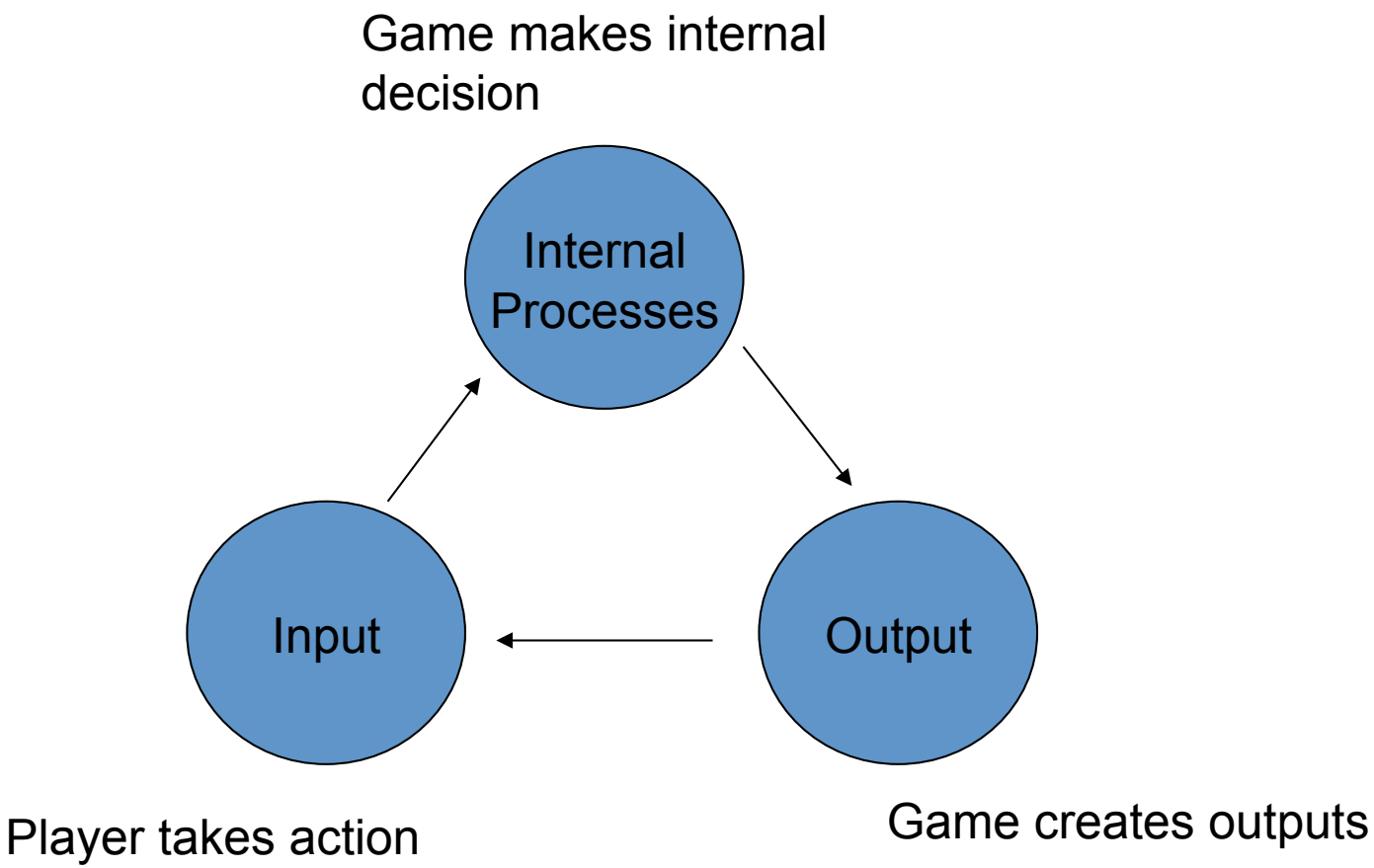
Tools for Balancing

- Design for Modification
  - Implement core rules
  - Configure rules with parameters
  - Store parameters in a modifiable form
  - Modify one parameter at a time, test game play
- Prototype well in advance
- Devise pay-off matrices
  - Look for dominant strategies

Now what?

- Now that we have our game design, how do we go about building it?
  - Complex interactive system
- We need to plan it otherwise it becomes a mess
  - Difficult to understand
  - Difficult to maintain
  - Difficult to extend

Interactive System



How do we put it all together?

- Inputs
  - Mouse, keyboard, controller
- Internal Processes
  - Evolving Game State
  - Objects, Rules, Procedures…
- Outputs
  - Graphics
  - Sound
  - User Interface

How do we put it all together?

- User interface
  - Configuration and selection
  - Help
  - Input / HUD
- Game Logic
  - Loading
  - Script
  - Physics Engine
  - Artificial Intelligence
  - Events
  - Collisions
  - Network communication
- Outputs
  - Graphics renderer
  - Sound and music

How do we put it all together?

- Game State
  - Position, orientation, velocity of all dynamic entities
  - Behaviour and intentions of AI controlled characters
  - Dynamic, and static attributes of all gameplay entities
    - Scores, health, powerups, damage levels
- All sub-systems in the game are interested in some aspect of the game state.
  - Renderer, Physics, Networking, and Sound systems need to know positions of objects
  - Many systems need to know when a new entity comes into or goes out of existence
  - AI system knows when player is about to be attacked – sound system should play ominous music when this happens

The Game Loop

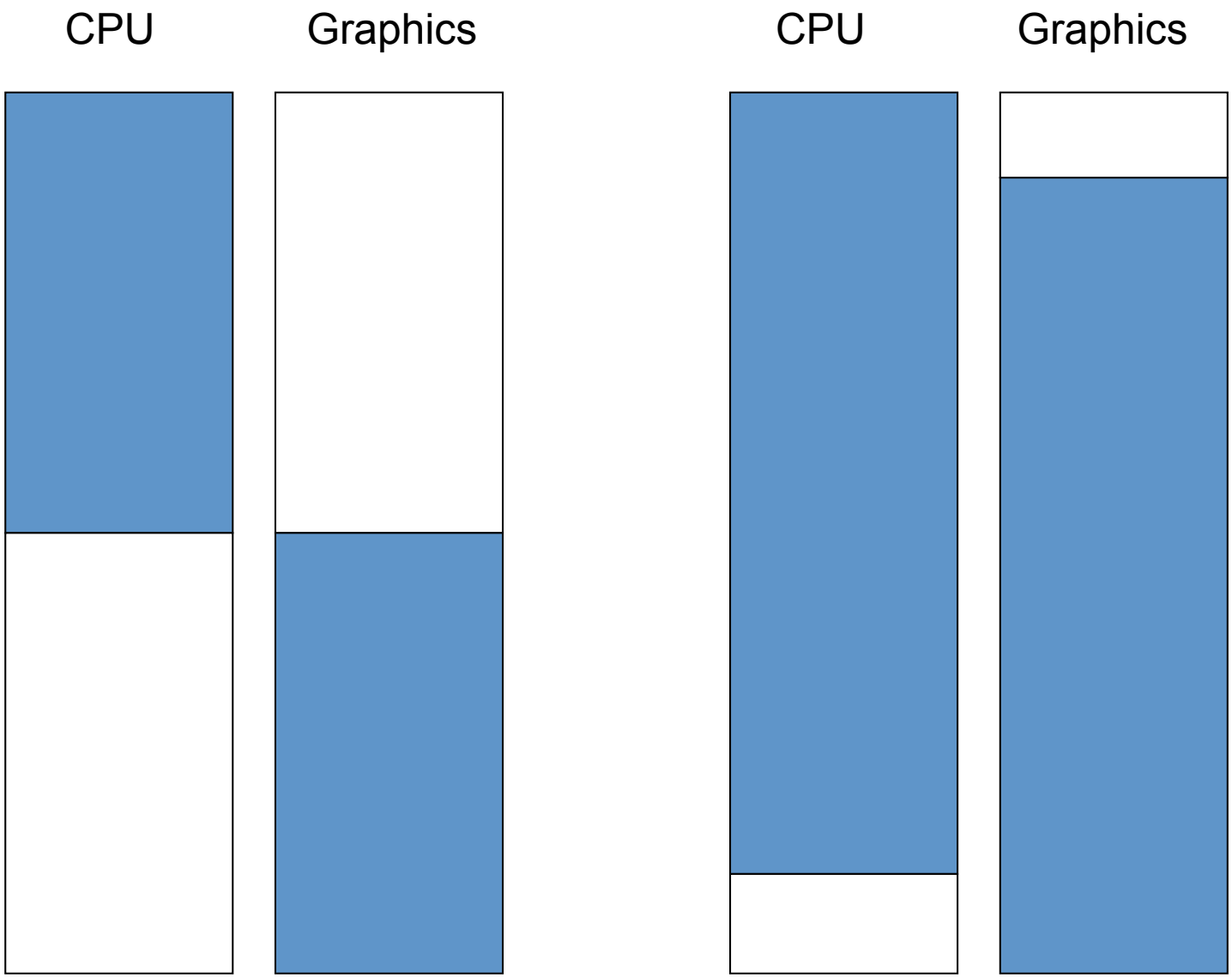
- The “heart beat” of a game
- Performs a series of tasks every frame
  - A series of frames are perceived as movement
  - E.g. 60 frames per second
- Run as fast as we can
  - A smooth game-play experience
- Potentially decouple to avoid bottlenecks

The Game Loop

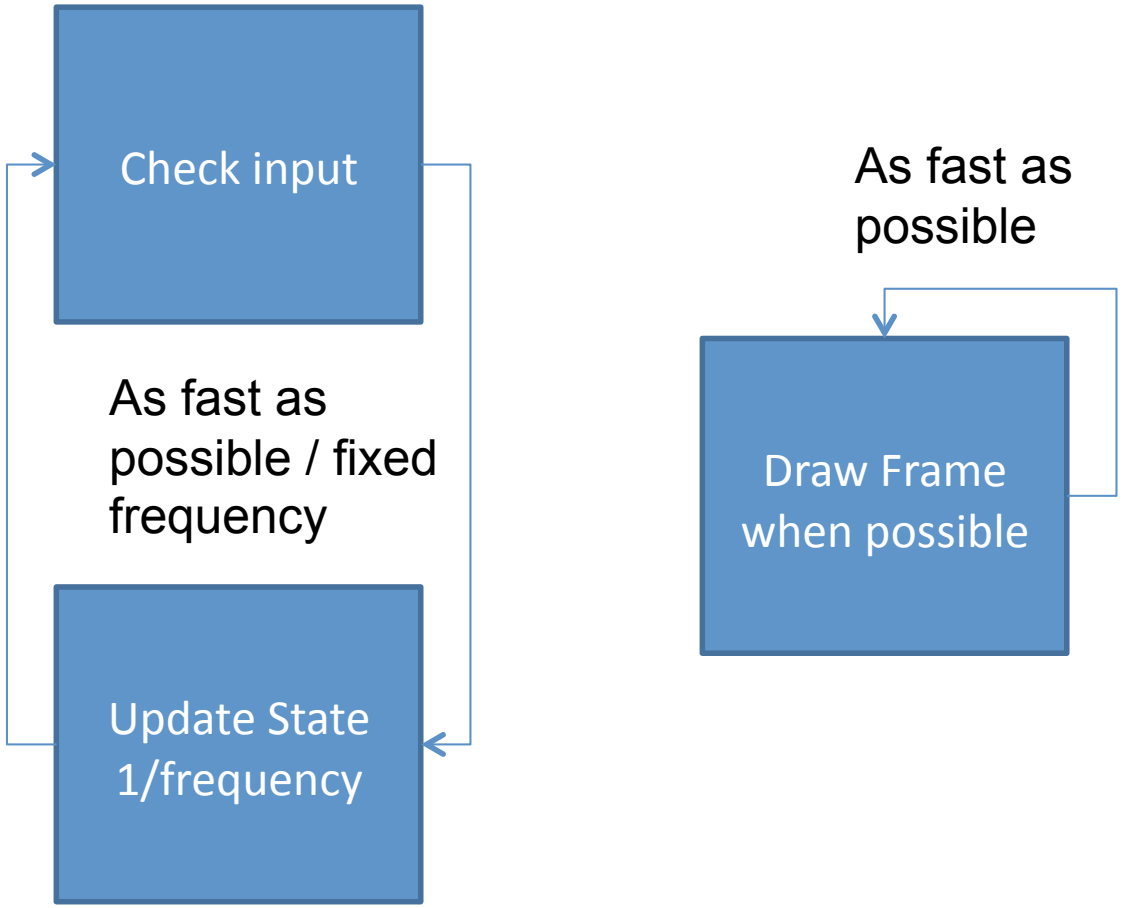
```
start game
while( user doesn't exit )
{
    get user input
    get network messages
    simulate game world
    resolve collisions
    move objects
    draw graphics
    play sounds
}
exit
```

The Game Loop

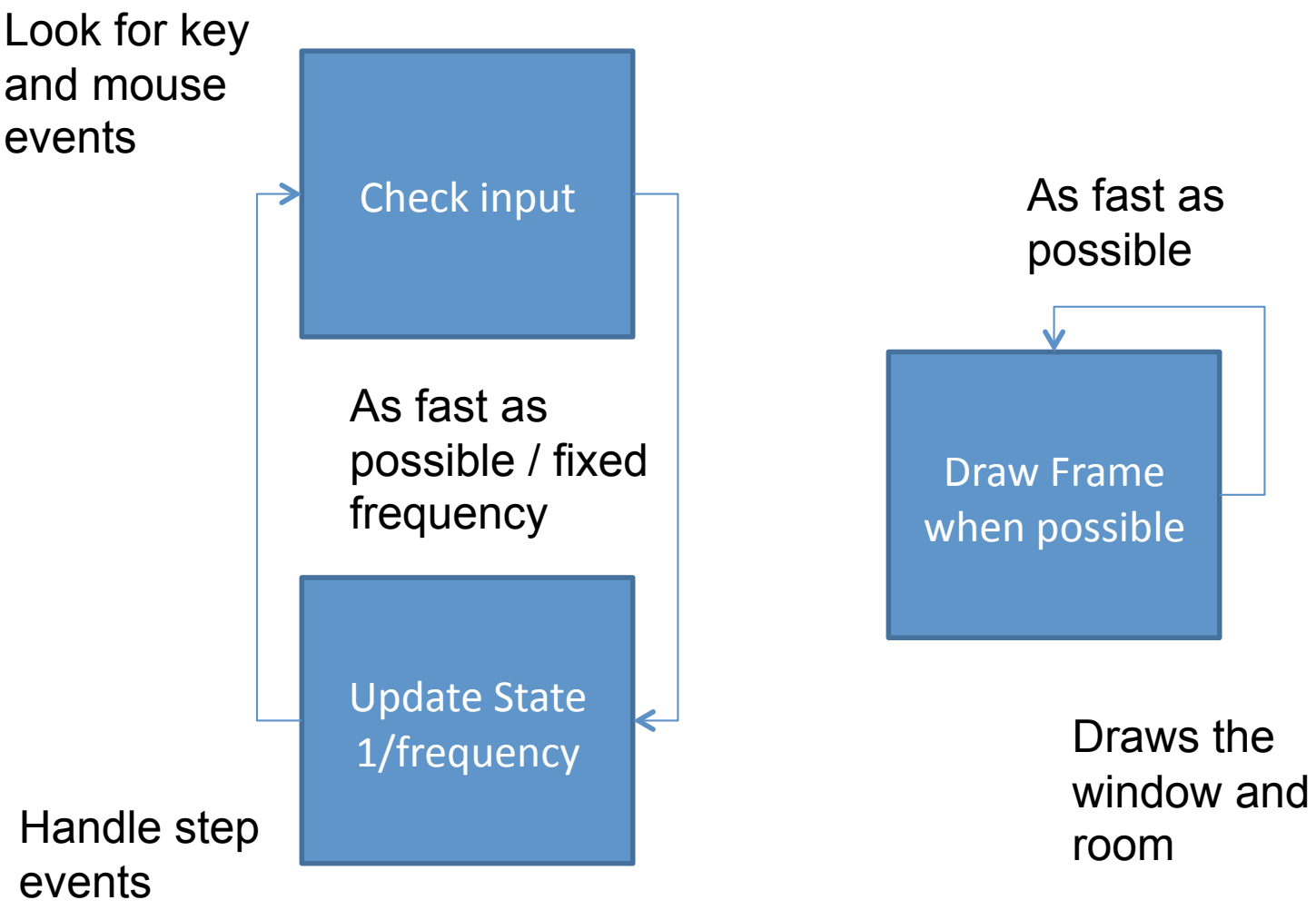
```
start game
while( user doesn't exit )
{
    how much time has elapsed?
    get user input
    get network messages
    simulate game world(elapsed time)
    resolve collisions
    move objects
    draw graphics
    play sounds
}
exit
```


CPU
Graphics
CPU
Graphics

Decoupling

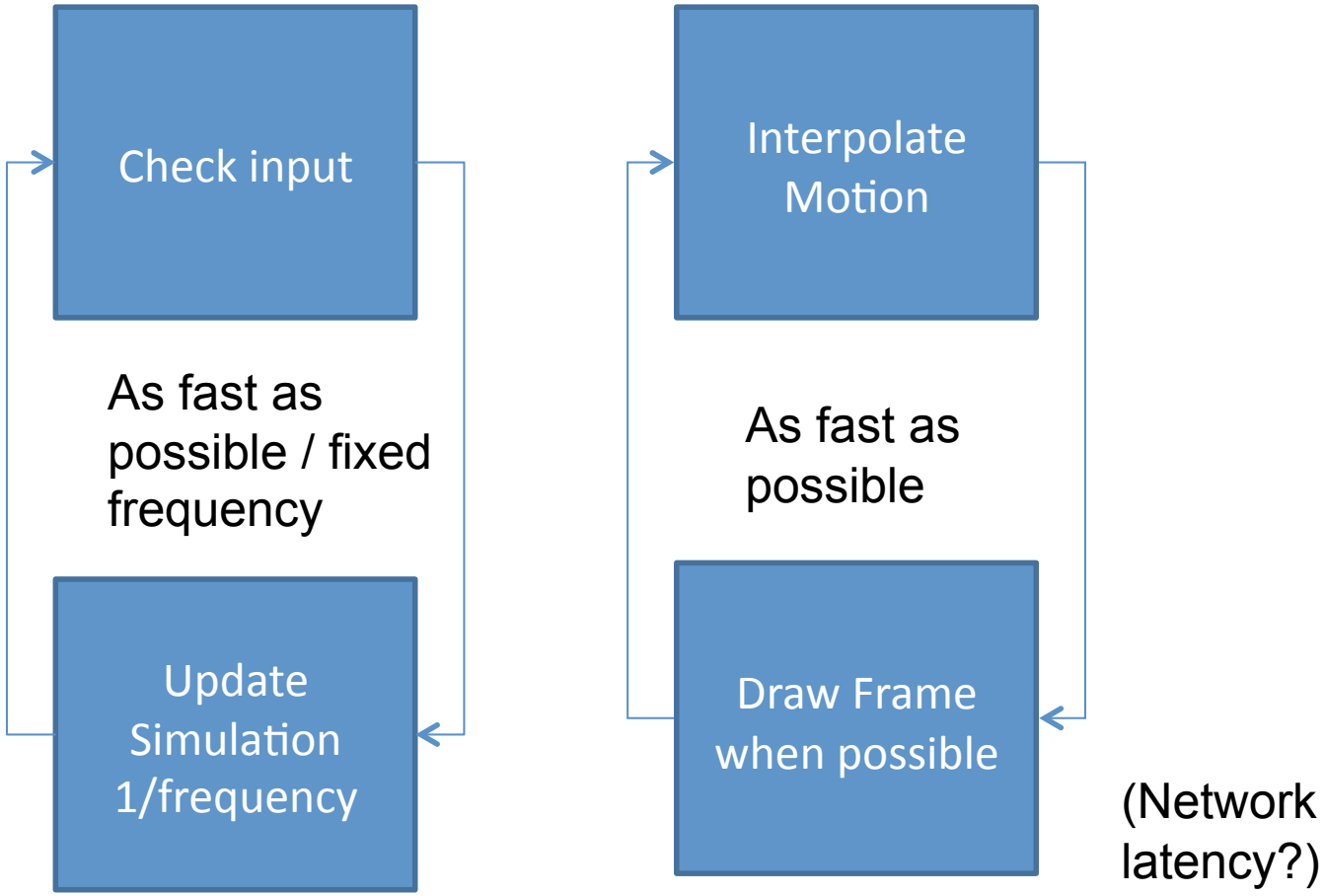

Check input
As fast as possible / fixed frequency
Update State 1/frequency
As fast as possible
Draw Frame when possible

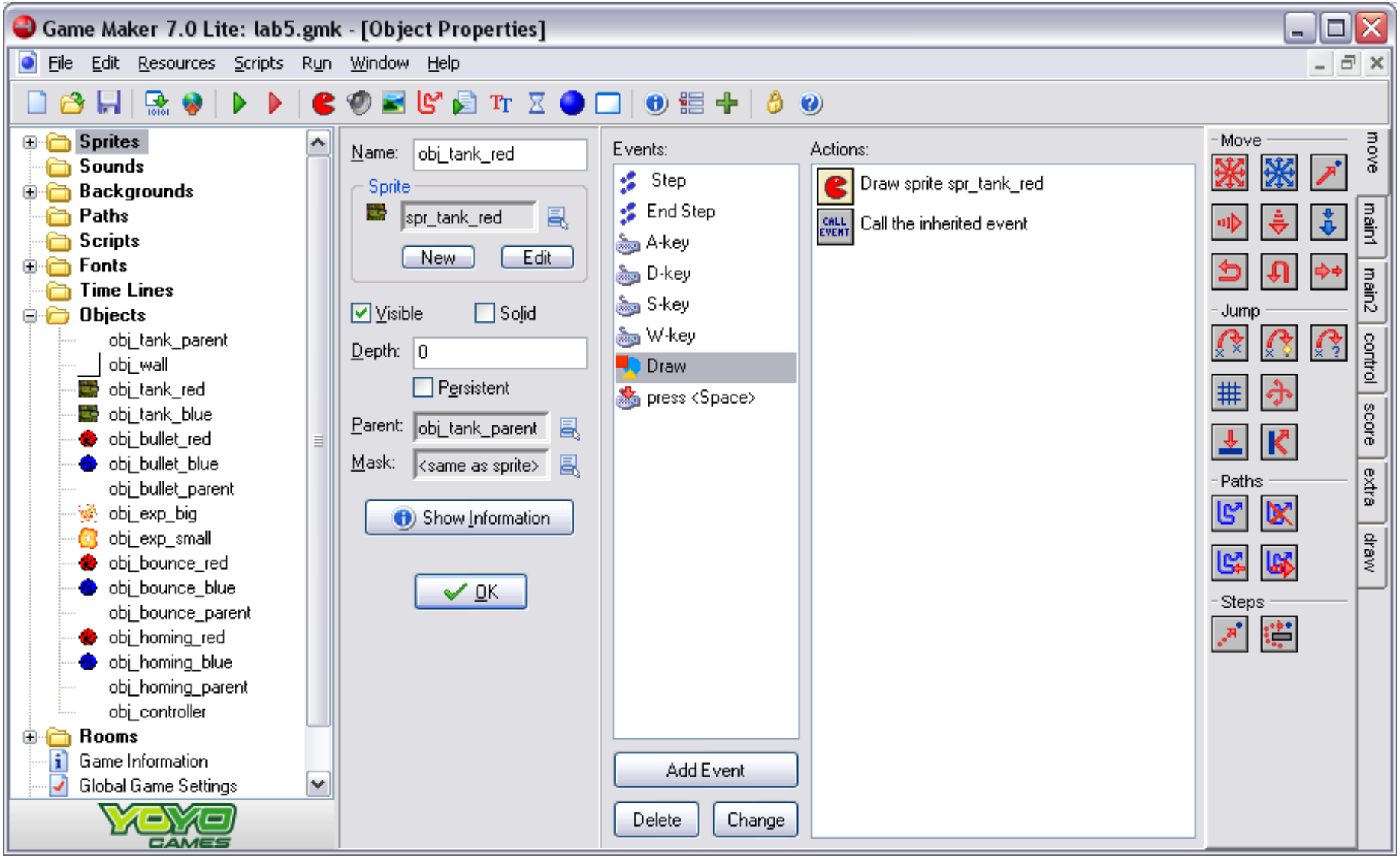
Decoupling


Look for key and mouse events
Check input
As fast as possible / fixed frequency
Update State 1/frequency
Handle step events
As fast as possible
Draw Frame when possible
Draws the window and room


GHOST TEAM

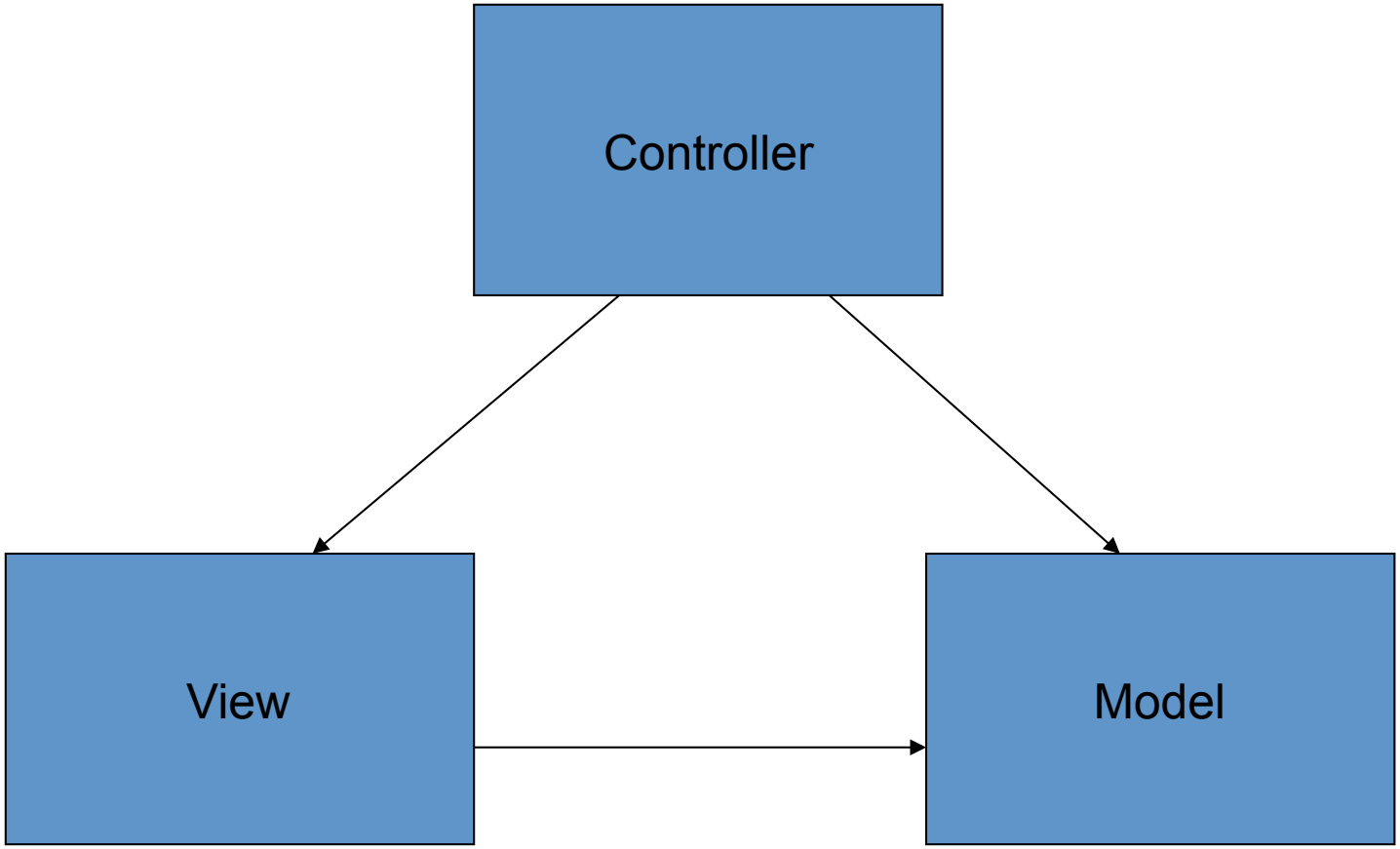
Decoupling


Check input
As fast as possible / fixed frequency
Update Simulation 1/frequency
Interpolate Motion
As fast as possible
Draw Frame when possible
(Network latency?)


Game Maker 7.0 Lite: lab5.gmk - [Object Properties]
File Edit Resources Scripts Run Window Help
Sprites
Sounds
Backgrounds
Paths
Scripts
Fonts
Time Lines
Objects
obj_tank_parent
obj_wall
obj_tank_red
obj_tank_blue
obj_bullet_red
obj_bullet_blue
obj_bullet_parent
obj_exp_big
obj_exp_small
obj_bounce_red
obj_bounce_blue
obj_bounce_parent
obj_homing_red
obj_homing_blue
obj_homing_parent
obj_controller
Rooms
Game Information
Global Game Settings
Name: obj_tank_red
Sprite
spr_tank_red
New
Edit
Visible
Solid
Depth: 0
Persistent
Parent: obj_tank_parent
Mask: <same as sprite>
Show Information
OK
Events:
Step
End Step
A-key
D-key
S-key
W-key
Draw
press <Space>
Actions:
Draw sprite spr_tank_red
Call the inherited event
Add Event
Delete
Change
Move
Jump
Paths
Steps
move
main1
main2
control
score
extra
draw

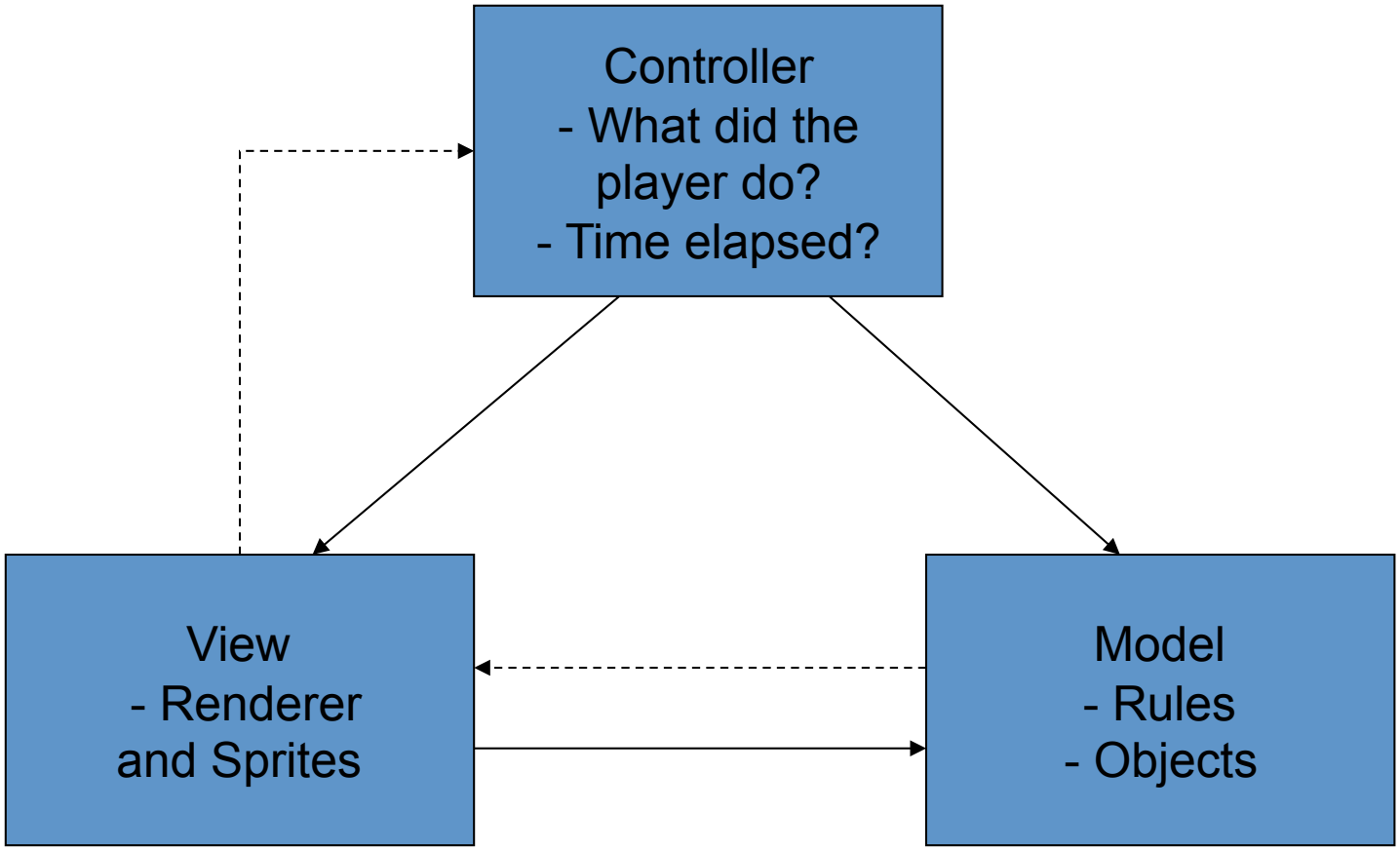
Model-View-Controller

- An architectural design pattern
- Used to isolate logic from user-interface
- Model
  - The information of the application
- View
  - The user interface and display of information
- Controller
  - Manages the communication of information and manipulation of the model


Controller
View
Model

Game MVC Architecture

- Model
  - The state of every game object and entity
  - The rules of the game world
  - The physics simulation
  - Knows nothing about user input or display
- View
  - Renders the model to the screen
  - Uses the model to know where to draw everything
- Controller
  - Handles user input and manipulates the model


Controller
- What did the player do?
- Time elapsed?
View
- Renderer and Sprites
Model
- Rules
- Objects

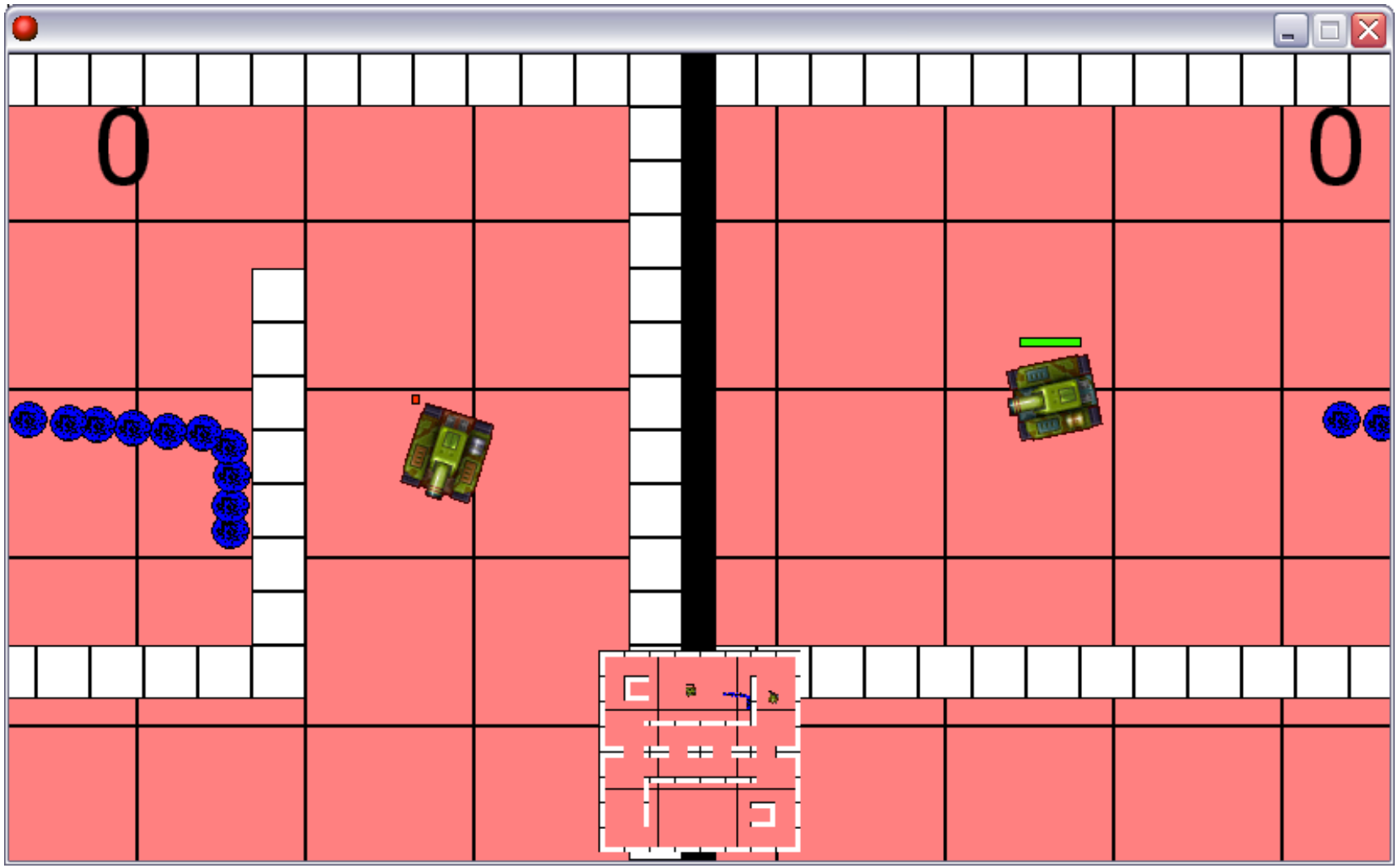
Quake MVC Architecture

- Model
  - An abstract 3d environment
  - Positions and orientations change over time
- View
  - Render the 3d environment
  - Display complex avatars and animations
  - Fancy effects
- Controller
  - Tell the model that I want to move, shoot, jump
  - Tell the model that 1/50th of a second has elapsed


Player
-(x,y,z)
-velocity
-Klesk avatar
-Railgun
-No ammo
-Bounding box

Why MVC is popular / good

- Nice modular design
  - Decouple game design from renderer
- Game world logic is bundled in the model
- Changes to the renderer / graphics do not affect the rest of the game
- Easily supports different input controllers and/or bots and AI
- Helpful when we think about networked games


0
0

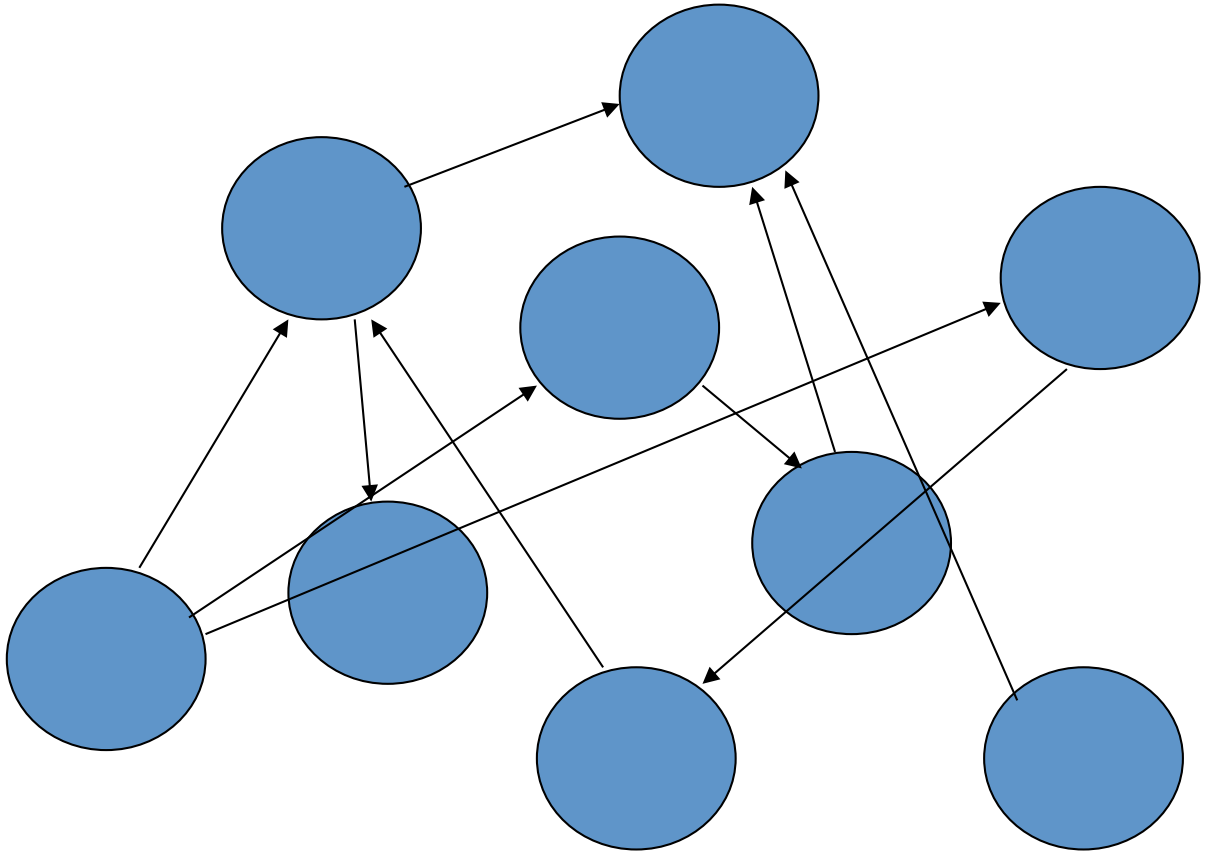

Gamer's Hell

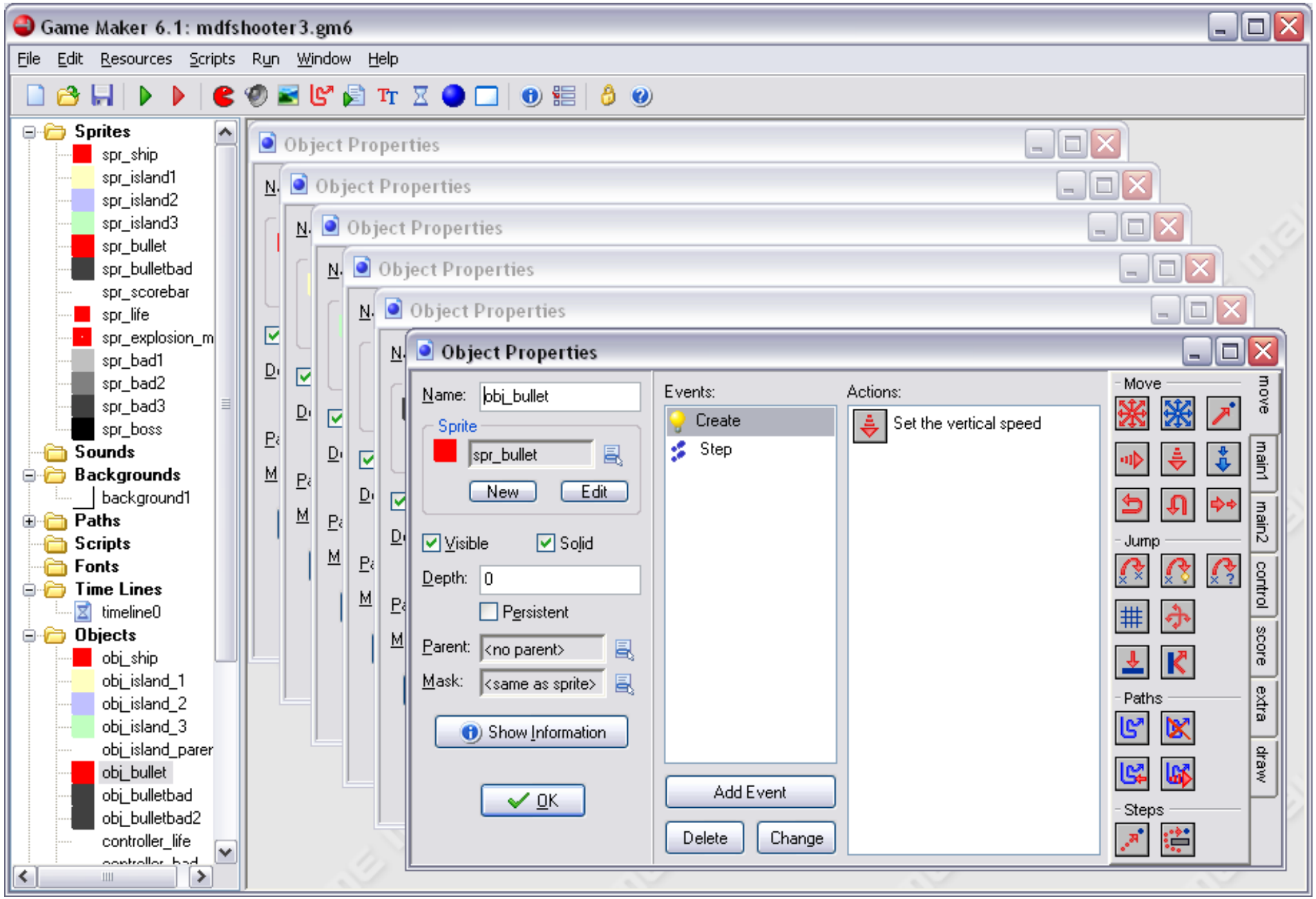
Structuring the Model

- Model
  - Objects
  - Rules
  - Together create game world and state
- Objects need to communicate with one another
- Objects need to be able to do things by themselves
- How do we structure this sensibly?

Direct Communication

- Object A attempts to pick up object B
  - Check if B can be picked up
  - Which functions A must call in B to reflect pick-up
  - Many conditional statements
- Bullet hits player
  - Who destroys the bullet
  - Who destroys the player
  - Who updates the score and the health?
- Why is this a poor design choice?
- Exponential complexity
- Every object needs to know how to interact with every other object




Game Maker 6.1: mdfshooter3.gm6
File Edit Resources Scripts Run Window Help
Sprites
spr_ship
spr_island1
spr_island2
spr_island3
spr_bullet
spr_bulletbad
spr_scorebar
spr_life
spr_explosion_m
spr_bad1
spr_bad2
spr_bad3
spr_boss
Sounds
Backgrounds
background1
Paths
Scripts
Fonts
Time Lines
timeline0
Objects
obj_ship
obj_island_1
obj_island_2
obj_island_3
obj_island_parer
obj_bullet
obj_bulletbad
obj_bulletbad2
controller_life
Object Properties
Name: obj_bullet
Sprite
spr_bullet
New
Edit
Visible
Solid
Depth: 0
Persistent
Parent: <no parent>
Mask: <same as sprite>
Show Information
OK
Events:
Create
Step
Add Event
Delete
Change
Actions:
Set the vertical speed
Move
Jump
Paths
Steps
move
main1
main2
control
score
extra
draw