UNIT-2

Use of Technology

Dr. S.P. Khandait
Department of Information Technology
KDKCE Nagpur

Introduction

- Architectural design is the skeleton of a project
- It does not detail the technical design
- Token refers to conceptual architectural objects and the word object refers to programming objects.
- In programming one-to-one mapping of tokens and objects.
- Business application is a tool to complete tasks; it does not have an exact goal like game.
- Good game provides challenges whereas good business application reducing human effort for performing various tasks.

The state of the art

- Game industry support cutting-edge technology
- Fast processor, new graphics card or any cool peripheral devices is introduced in the market, game developer immediately give their support to it.
- Game developer need to keep the interface and control model same between ganres, the game's appeal lies in its novelty value.
- Games take a long period to develop and with sudden market changes, it becomes difficult to predict the future trend.

Technology perception

- Always counts first impression
- Most impression comes from reviewers in games and sport magazines
- Smashing effect on the player at first instant, chances of good score
- Game review gives game product life longevity
- Gameplay gets the highest priority followed by appearance and accessibility

- Adventure games are technology-led
- They use nice 3D engine, music, and most importantly the implementation of a real time inverse kinematics engine.
- For example, rotating the arm about the elbow or around the wrist until it reaches the correct position is forward kinematics.
- But when we need to move the hand to a position we want and correspondingly move the forearm, upper arm and shoulder t support, it is inverse kinematics.

Game research

- Game designer first gets an idea(vision) and visualize the basic game concept in his/her mind, and then needs to understand game subject matter and this phase is called is called research phase.
- Game research is essential to have an expert understanding of the game vision.
- Game professionals need documents that contain background information, object specification, images, texture, various character attributes and numeric information.
- Research documents educate the entire development team and aid the game design process.

Research goals

- Goal1:football field dimension
- Goal2: picture of stadium, photos of players, manager, coach
- Goal3: Picture of the jersey(home and away) along with players jersey number.
- Goal4: Team formation and team information of all the other in the league.
- Goal5: Videos and photos of players get their trademark shots, maneuvers and goal saves.
- Goal6: Team's playing schedule and method of determining the championship team and relegation teams.
- Goal7: Manager's information, his style of selecting players for the game and trading players, new contract and renewals with salary fixation.
- Goal8: Income through pre-sold tickets, selling jersey, sponsors ticket prices.
- Goal 9: Referee rulebook to understand penalties, warning, and offside/goal signals.
- Goal 10: Audio and video of stadium sounds, fans cheering , announcements, commentary, ball kick, specific goal celebrations of each player.
- Goal 11: Additional information about the clubs legendary players and their statistics

Research Sources :

1. Books and encyclopedias:
2. Magazines:
3. Newspaper:
4. DVD,CD, downloads:
5. Internet:
6. Games hint book:
7. Fan websites and blogs:

Blue-Sky research

- Research is essential for the growth and survival of a company.
- Blue-sky research can be dangerous if care is not taken, since it is an undirectional research.
- In this, the developer are free to research the area that interests them
- Blue-sky research projects are often chaotic and more of a gamble
- Research can be encouraged if it has an aim and it is the outcome of research, could not be done on optimizing and improving the known techniques, instead of finding a completely new method as in blue-sky research.

Research types

1. Research on existing product
   - First thing is to find out if any similar product is available in the market?
   - Find the positive and negative of the competitive products.
   - The marketing department can provide some useful statistics
   - We need to keep a watch on new releases in the genre
   - Product should excel in artwork(presentation), technology(performance) and game design(innovative) to stand in the market.
2. Target market
   - This is for sales prediction
   - Is the target market the mass market or is it specific(niche)?
   - Marketing strategy will depend on the type of market
   - For niche market, the fans have to be targeted
   - It is very difficult to get their acceptance unless the product is too good
   - To stay in market for quite some time before any competitor brings out a decently similar product for a low cost
   - this should not suggest that target audience is to be only a mass market to see profit
   - Sometimes, we can have a tie up with a hardware company to promote sales
   - By doing this both companies prosper
   - For example, release of Tetris on game boy. Tetris was responsible for almost 6 cores game boy sales worldwide

3.Research on Gameplay

- Extensive research to find details of various aspects of gameplay is needed
- Digging the history in that domain can be of help to provide more feature
- A strong history background can make the game interesting as well as educative
- The vision of the game can stimulate an idea for an invention
- The gameplay will include spatial, temporal and logical reasoning using he manipulation of abstract concept mapped to concrete entities
- All these will lead to the way in learning how the game rules operate
- The gameplay idea can be obtained from anywhere since information is available all around us
- The gameplay can also have an impact on the technology required for the game development

4. Research on Technology

- This research is to find the kind of technology that is required to implement the game
- The decision to follow any new technique has to be verified
- This is also requires approval for time and budget
- Management would want concrete result from research
- The research is an unpredictable activity and in new technologies, it is particularly very difficult
- New versions and releases will have refinements more on the gameplay than on technology
- Most single player game is also a multiplayer game with the computer controlling all the other players.
- This defines the conventional game theory
- There is no actual gameplay in the multiplayer games; it is more of a simulation in a fantacy environment
- Accurate simulation brings it close to reality

Research Journal

- Research have to be documented from the thought to every procedure and its results both incorrect and correct ones.
- Research is needed to keep up with the fast growing technology, otherwise we will left behind in the competition
- Every member of the research group has to document all the steps that they have done
- This is also quantifying the amount of effort that has gone through the research finding
- This can provide improvement and suggest modifications to the gameplay design
- The most important part of research is not the result
- It is knowledge and experience one gains through research

Reinventing the wheel

- All application developers look for third party libraries
- Now a days, lot of third party libraries available for graphics, 2D engine 3D engines, physics system, audio routine, sound, music, digitized voice libraries and source code, FMV player, installation procedure etc
- Primary reason for using game engine is to give the content creator more time to work on the title
- Advantage of using a licensed game engine because it is easier to hire people who are already familiar with it
- Game engine has to be platform independent

Use of object technology

- Until the last decade, object oriented techniques are not used by gaming industry
- OO languages were very slow since they produced a lot of additional code behind the scene to handle the OO nature of the language
- The virtual functions were slow because they required an extra address lookup

Myths

i. Java is too slow to be used for game programming
   - True with JAVA 1.0
   - It was 20-30 times slower than C/C++
   - Java 5 an average is about 1.1 times slower

ii. Java is too high level
   - Java has a large class libraries and so implementation is more easier in java

iii. Java is rarely used to write real games
   - Not true
   - Well known games like star wars galaxies, puzzle pirates, galantic village and many freeware games were written in java

Algorithm optimization

- Optimization is of two types: algorithm optimization and code optimization
- Algorithm optimization is preferred to code optimization
- For example, use of quick sort instead of bubble sort algorithm
- We need to test both algorithm with the input data and find whether the performance is close to our expectation
- input list contains, two classes of object corresponding top static and dynamic tokens
- Full sorting needs to be performed only a few times, namely when the list is created and when the viewing angle changes; not when every frame is drawn
- i.e the need is that the list is to be created only once for every level and the static objects do not move and so are sorted only once
- With dynamic tokens, z coordinates changes as they move and so their position in the list is to be modified
- Insertion sort using for dynamic tokens in every frame since the number of dynamic tokens are very small when they compared to static tokens

- At the beginning of each frame, the dynamic tokens insert themselves in the correct position and at the end they are removed from the list
- one of the drawbacks of this technique known as "Painter's Algorithm"
- This means that when we start drawing the farthest object to the camera, the closest once overdraw those behind. Therefore the algorithm is insufficient

- The solution to this is to implement token based z buffer
- This solution reverses the order of the sort which means the tokens closest to the camera were drawn first causing the z test to fail with tokens that were away and obscured
- This reduced the average time to draw the frame
- Final conclusion good optimization can be achieved only by looking at the data structure and choosing an appropriate algorithm

The pros and cons of abstraction

- Abstraction separates interface from implementation
- Changing the sort method involved only one change in the line of code to instantiate another class and recompile
- Disadvantage of abstraction is that it is hard to do good abstraction
- It adds more overhead to a program
- Therefore some moderation is required
- Concept of granularity needs to be understood which will be dealt with in the next chapter

Building Block

- Introduction
  - Software factory refers to a methodology of producing software that centralizes and simplifies the production of specific common modules
  - These common modules form a core set of tools and libraries and are well maintained and supported over a series of products
  - Software factory method is well suited for project with common functionality
  - From management perceptive, this saves a lot of money by using the already written code instead of writing them once again by specialists.
  - These codes are already tested, integrated and debugged

The tasks which can be made into reusable modules generally are:

1. Screen code
2. Sound setup code
3. CD track playing code
4. Data file loaders
5. Compression and decompression libraries
6. Windowing and graphic features
7. Menu code
8. Environment setup
9. Hardware, software configuration
10. Encryption/decryption code

Advantages of software factory

- More code reuse can reduce the project development time
- Since the code is tried and tested, it is more reliable
- The efforts of skilled programmers are reduced and they can be used for doing other tasks
- It also spreads knowledge about those modules to many developers
- Project progress can be easily seen

Disadvantages of software factory

- The code has to be more generic to be used across different products
- It takes more time and effort to develop
- The first project which makes reusable modules takes longer time
- For each platform, separate reusable wrapper libraries must be developed
- These libraries have to be maintained and new developers have to spend time learning these new libraries

Game development issues

- Common issue in game development are platform independence and the risk in losing key personal

Platform independence

- In the architecture design, we need to provide interface to all the platform-specific codes
- Every platform of different h/w configuration with different graphics card, sound card, etc
- These codes have to be written and linked through a common interface
- Using middleware we can solve the problem of hardware independence
-

Risk reduction

Core Groups in Software Factory & their Interaction

1.Game design group

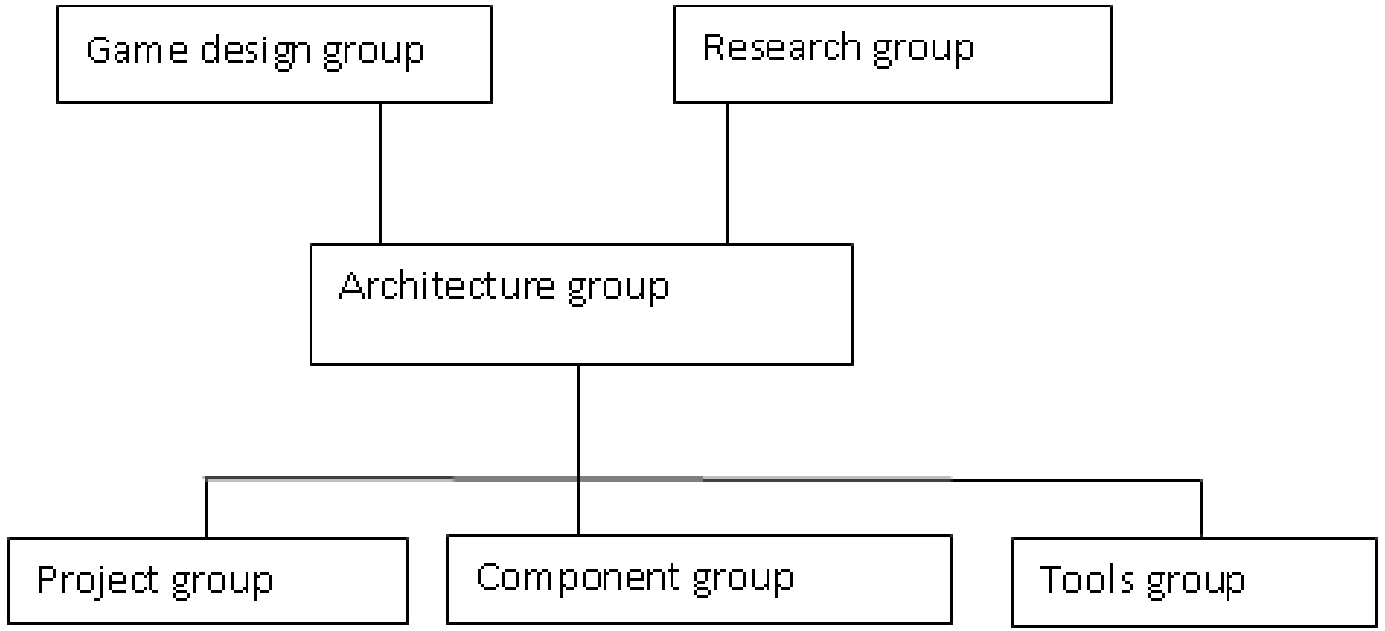
2.Architecture group

3.Tools group

4.Project group

5.Research group

6.Core component group



Game design group

- The game design group mainly interacts with the architect group to know the project visibility (progress)
- They can provide input to tools and components group
- For example, they can directly contribute script to AI scripting engine or if a database is developed to store tweak able parameter, then game designers can provide more parameter to it
- These parameters are part of soft architecture and can be modified without affecting the hard(structure) architecture
- The game designer group also interacts with ancillary sound and art group to specify the look and feel of the game

Software architect group

- The software architect group acts as hub between game designer group and the other groups
- The members of the software architect group acts as a leader to all the programming group
- The core function of this group is to translate the game design document into technical design document understanding the available tools and components
- If necessary tools are not found, they prepare specification for modifications of components and tools or initiate development of new components and tools after research
- Members of this group provides the feasibility report to game design group and also provide advice regarding the technical issues of the game
- This group is also responsible for keeping control over company standards for coding and documentation
- These standards include file formats, source control standards, interface definition standards, directory structures, etc
- Members of this group need high technical skills and should be able to answer any technical query from all other groups

Tools group

- Primary responsibility of this group is production and maintenance of tools that will be required by all development groups
- Some tools are built in house, while there are some off-the-self tools available in the market like 3D studio MAX

components group

- Responsible for developing low-level modules, interfaces to other third-party components, platform-specific libraries and modules related to some common tasks such as compression libraries
- As experience of the team grow more functionality can be added
- Each component should have a an interface so that implementation changes do not have an impact on other modules and perform only one task to allow reusability

Project group

- Prime goal of this group is to find the feasibility of the project
- For this they have produce prototype of the product with the available support from component and tools
- If in the prototype no anomalies found, then project gets a go ahead
- The game design group interacts with project group through the architecture group to ensure that project schedule is maintained

Research group

- Main function of this group is to investigate and prototype new technologies and include it in the libraries for use by all projects
- The results of their research are detailed in journal and this journal is frequently reviewed and checked
- The purpose of this group is to remove all open ended research from the critical path of development so that the risk of schedule slips does not arise
- The size of this group varies depending on the requirement of members in other group

Ancillary group

- Sound, art, testing, marketing and management belong to this group
- Though directed by architect group this group can directly interact with game design group
- Main function of testing group is to do integration testing compatibility testing

Summary of core group interaction in software factory

- ✓ Assemble programming group for each project according to its needs and make efficient use of resources
- ✓ Understanding the strength and weakness of each group. Do not over burden them
- ✓ Allow knowledge transfer by moving the free members to another group which has more work
- ✓ knowledge can be spread among team members by rotating them frequently to different groups except when the project is nearing completion
- ✓ The knowledge transfer also minimizes the risk if a member leaves the company all of sudden

Design reuse

- Design pattern is a solution for a problem that frequently come up in software development
- Design patterns arise out of experience of designers and are easily understood and applied to various projects by developer
- These are specific to domain and so that developer are understanding the domain well before they start coding
- Some common design patterns applicable to games are given as follows:

Design reuse

1. Object factory
   - Object factory creates a family of object
   - All the object created by the factory are derived from the same abstract class and are returned as reference to this class
   - The object factory keeps track of all these objects in a list
   - When an object is deleted, it is removed from the list
   - Object factory receives request from a client to create an object representing a token within a level
   - Each token has a unique ID and loading is passing that ID to object factory and getting a return pointer to the base class
   - At the end of each level if object remains in the list, they are automatically deleted
   - This is done using a memory tracker class, which is an instantiated member of the game object class
   - The constructor of this class stores a reference to game object and inserts itself to the list in the factory class
   - When a game object is deleted, the destructor member of memory tracker is invoked which removes the reference to itself from the factory class list

Design reuse

2. Singleton pattern

- This pattern ensures that only one instance of a particular object can exist in the application
- Ex. Only one instance of an object wrapping the sound card or graphics card is required
- This singleton object is instantiate mostly on program initialization or on first use of the object
- In the first case, the constructor of the singleton is first called before the main function and in the second case , it is called on demand
- Singleton not instantiate if not called
- The static modifier takes care that only one instance of singleton class exists
- An example would be for a class that read and writes a configuration file

Design reuse

3. Flyweight pattern

- This pattern is used along with singleton pattern'
- The flyweight allows multiple instance of the class to be instantiated, but all of them refer to a common component shared by them all
- Individual configuration data is stored within flyweight object
- Advantage of this pattern is that it saves memory
- Disadvantage is it can be used only for read only resources
- It is of no use if local modification to the global resource has to be made

4. Chain of responsibility pattern

- This pattern set up a chain of objects that receive notification of an event thereby avoids direct coupling of event, signaling the object with the chain of object interested in handling that event
- Implementation can be through class diagram or runtime object hierarchy
- Disadvantage of class hierarchy is that the chain is fixed at compile time, meaning the class hierarchy can not be changed

5. Iterator and reverse iterator pattern

- This pattern allows the game developer to simplify the algorithm that needs to iterate on a list of items, so as to increase the speed of game execution
- In game development, we could create an iterator that will iterate through a list of game objects and return them in the ascending order of their distance from the camera
- The code will contain the class declaration for the list with *Insert()* and *Get()* members to insert and retreive members
- An abstract class iterator is defined outside the list class declaration
- This defines the interface for an iterator and allows iterator for all collection classes

6. Templates and strategy methods pattern

- This pattern allows dynamic selection of algorithm
- Ex: a computer-controlled character could use the strategy method for choosing different styles of play
- Coder can implement a generic style and attach a new strategy at runtime to customize the style
- Later, a different styles can be added if required to the generic class encapsulating a new algorithm in concrete class for that style of play
- Adding many styles can create a problem in managing and maintaining code

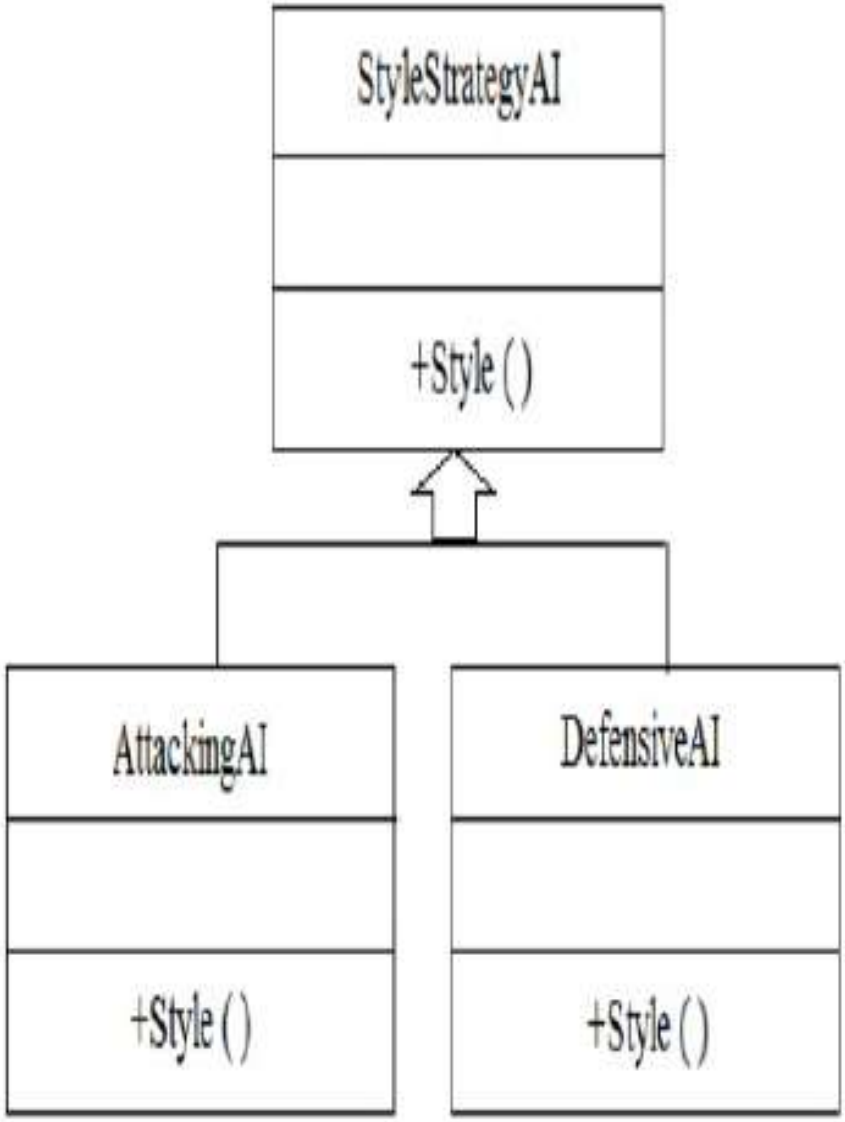


Fig: Class diagram for strategy pattern

7. Observer pattern

- Also called as "publish-subscribe model"
- One to many relationship between observed and observer object
- When observed object changes , all the objects are notified about that change
- Disadvantage of this pattern is that the object being observed has no way of finding out which are the objects observing it
- This can be achieved by allowing each observing object to subscribe to main object that is being observed
- This gives the possibility of applying the filtering condition at source to prevent too many notification messages being sent out

8. Command pattern

- Allows commands to be encapsulated and passed around as objects
- In games, AI is implemented by using scripts
- By scripting AI, compiler time is saved
- Command pattern encapsulates game functionality as commands which can be typed into a console, stored and replayed or even scripted to help test the game
- To implement this, the scripts are written in the text file and used along with an object factory to create the command objects specified in the text file and then link them

9. Facade pattern

- This pattern provides a simplified interface to third party libraries or a subsystem of related classes
- This reduces the complexity of using the subsystem interface
- The façade pattern also enables portability of code across different platform
- i.e. the same client source code can be used across different system libraries if the façade interface is implemented correctly on all target systems

10. Mediator pattern

- It is very close to the façade pattern
- The facade works as an interface between a client and a set of subsystems, whereas a mediator manages the interactions among the subsystems
- Communication in façade is one-way, but the communication in mediator it is two-way
- Advantage of this pattern is that it promotes loose coupling by removing the need for the mediator objects to hold references to one another
- The mediated objects have been to conform to the interface rules of mediator
- The mediator implements as an observer, because it observes the mediated object and responds to events raised by them

11. State pattern

- allows an object to change its behavior according to the changes that happens in its internal state
- Ex. In the fighting game the fighter character can be in one of the three states namely, “angry”, “very angry”, “extremely angry”
- In each of these states, the fighter uses different set of moves

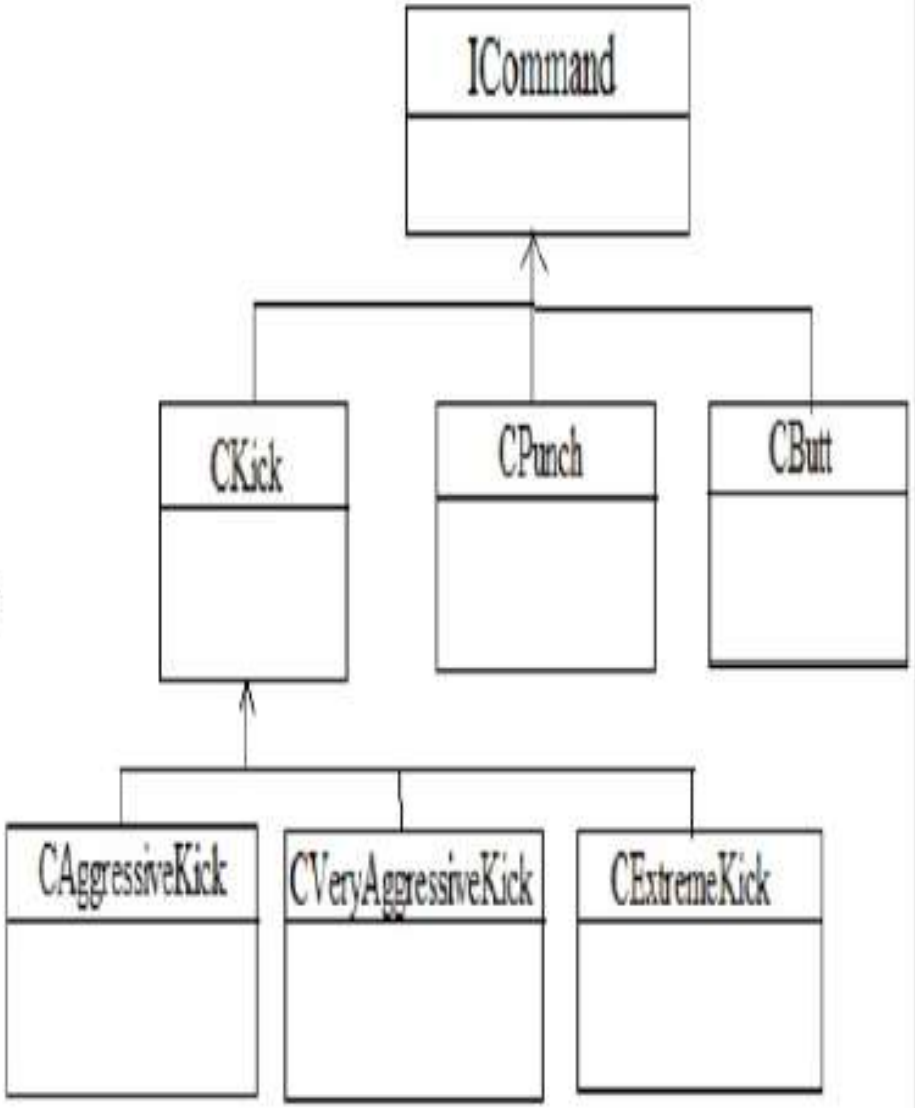


Fig: State pattern for fighting game

Design Reuse summary

- Design pattern offers general solution to problems
- Not all the patterns are useful for games
- At the architectural level, all games of a genre are inherently similar
- Using design pattern definitely reduces the effort and thereby the cost of development

Initial Architecture Design

Introduction

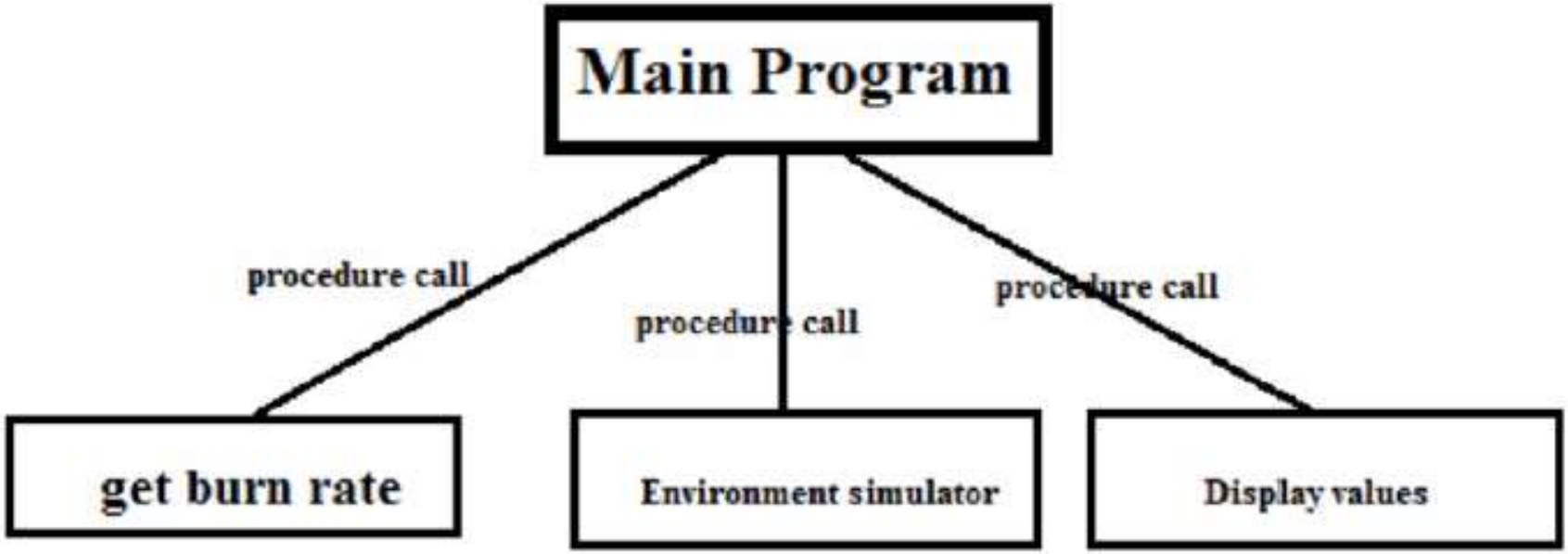
- Architecture begins at the outset of any software activity and not a phase of development
- The solution (design) and structure (architecture) are equally important for any product development.
- Architecture refers to the conceptual design of software development, the key abstraction that form the initial design.

Architectural Styles

- Architectural style are basically lessons learnt from experience in software system design.
- The Architectural style reflect less domain knowledge than architectural patterns, and hence are more broadly applicable.
- It is a collection of architectural decisions that are applicable in given development context along with the constraints specific to a particular system within the context.
- The resulting system elicits the beneficial qualities of the architectural design.
- The most popular architectural style are given as follows.

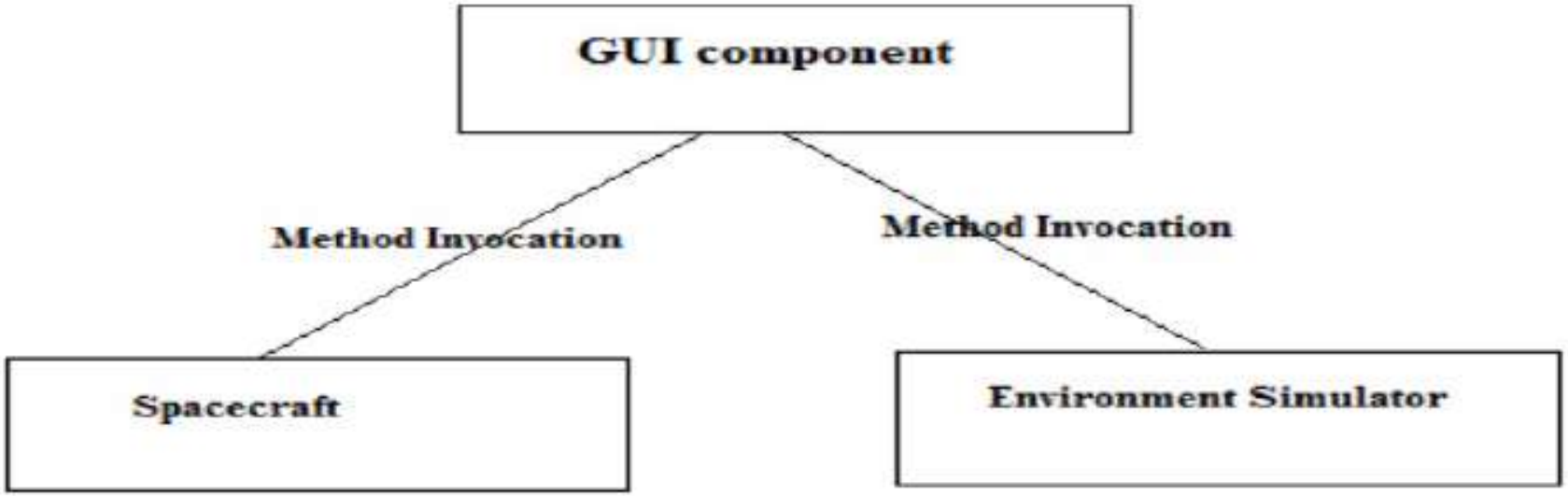
Traditional language based style

- Main program and subroutine are component and procedure calls are connector
- Data elements are passed in and out of subroutines
- Benefits of this style is modularity
- Small programs can use this style
- Ex: 1st component gets throttle setting(burn rate) input from user for pilot. 2nd is simulator that translates throttle setting i/p into burn rate in order to control the engine. 3rd display updated state.



Object oriented style

- Objects that encapsulate functions and data form the component
- Objects are instantiated before methods are called
- Method invocation (procedure call that changes the state of the object) is the connector
- The arguments to these methods are data elements
- The benefit is the integrity of data operations
- Applications involving complex, dynamic data structures can use this style
- Three components here are GUI, spacecraft, and environment simulator



- Layered Style
  - Here architecture separated into ordered layers
  - A program or component in on layer can obtain services from the layer below it
  - It could be in virtual machine style or client server style
  - This style used for developing operating systems and network protocol stack

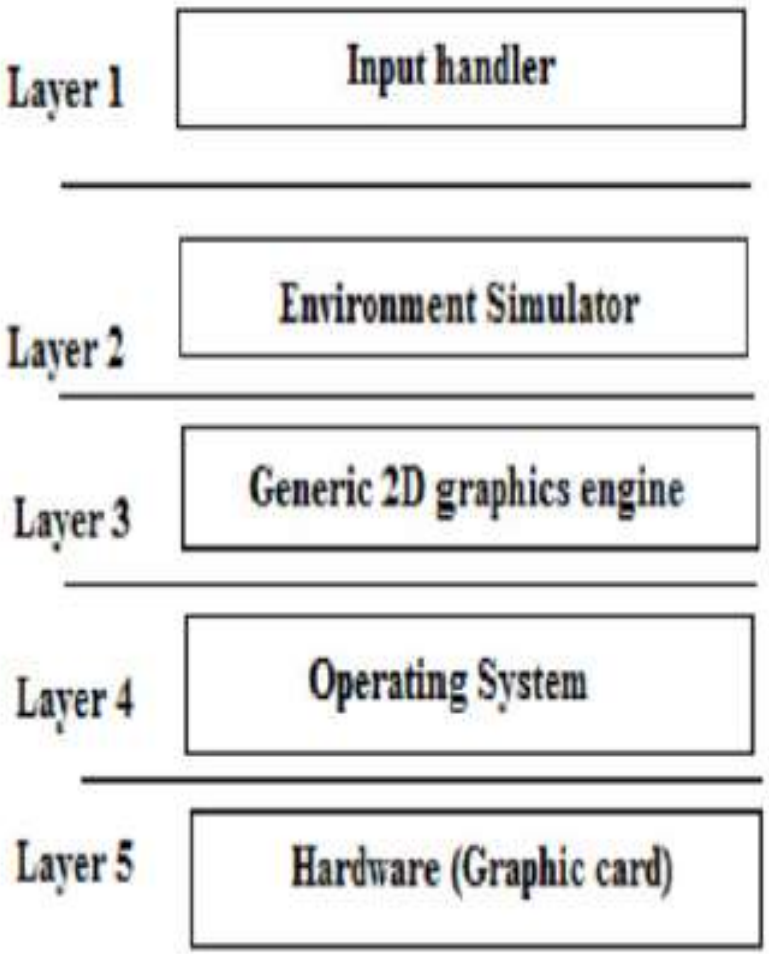


Fig: Virtual machine style for Lunar Lander game

Client Server style

- It is similar to virtual machines, but contains only two layers with network connections
- Clients makes request to the server via remote procedure calls(RPC)
- Multiple clients can access the same server; and server provides the requisite service for each client
- Client are mutually independent
- If a client performs more than user interface functions, it is called a "thick" client, else known as a "thin" client

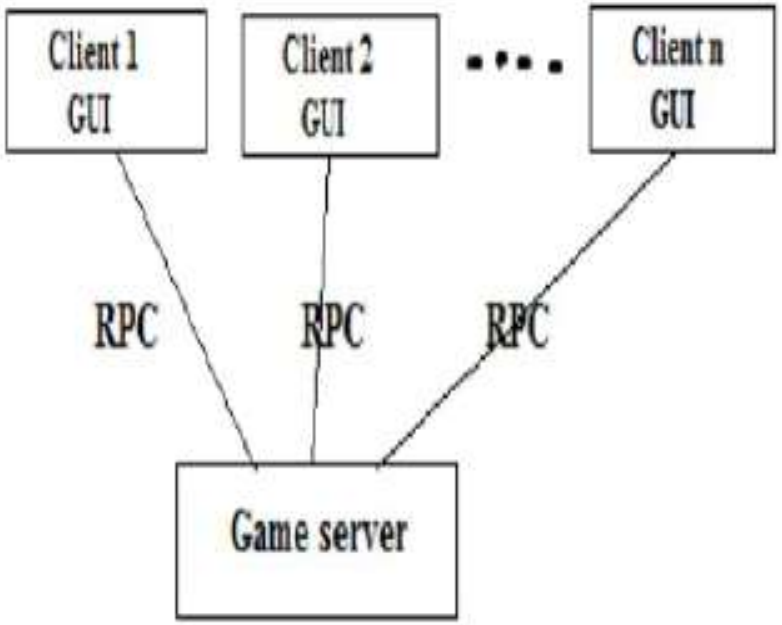


Fig: Client-Server style for Lunar Lander game

Data flow style

- This style concerned with the movement of data between processing elements
- This includes batch sequential and "pipe and filter" styles
- The batch sequential style mostly used in business application where some processing has to be done on large datasets, and is not suited for most games development
- In pipe and filter style streams of character data are passed from one filter program to another

Fig: Pipe-and filter style for Lunar Lander game

Shared style

- Also known as rule based/expert system style
- Shared memory is a knowledge base, i.e. a database with facts and production rules which consists of "if...then" clauses over a set of variable
- The user can input facts and production rules, and also query the database
- An inference engine operates on the knowledge base in response to the user input
- In interface engine parses the user input, and if it finds the input to be a fact or production rule then it adds that to the database
- Benefits of this style is that rules can be added or deleted from database dynamically
- for lunar lander problem, the rule based style requires a database of all facts about the spacecraft
- The physics model can determine the state of the spacecraft

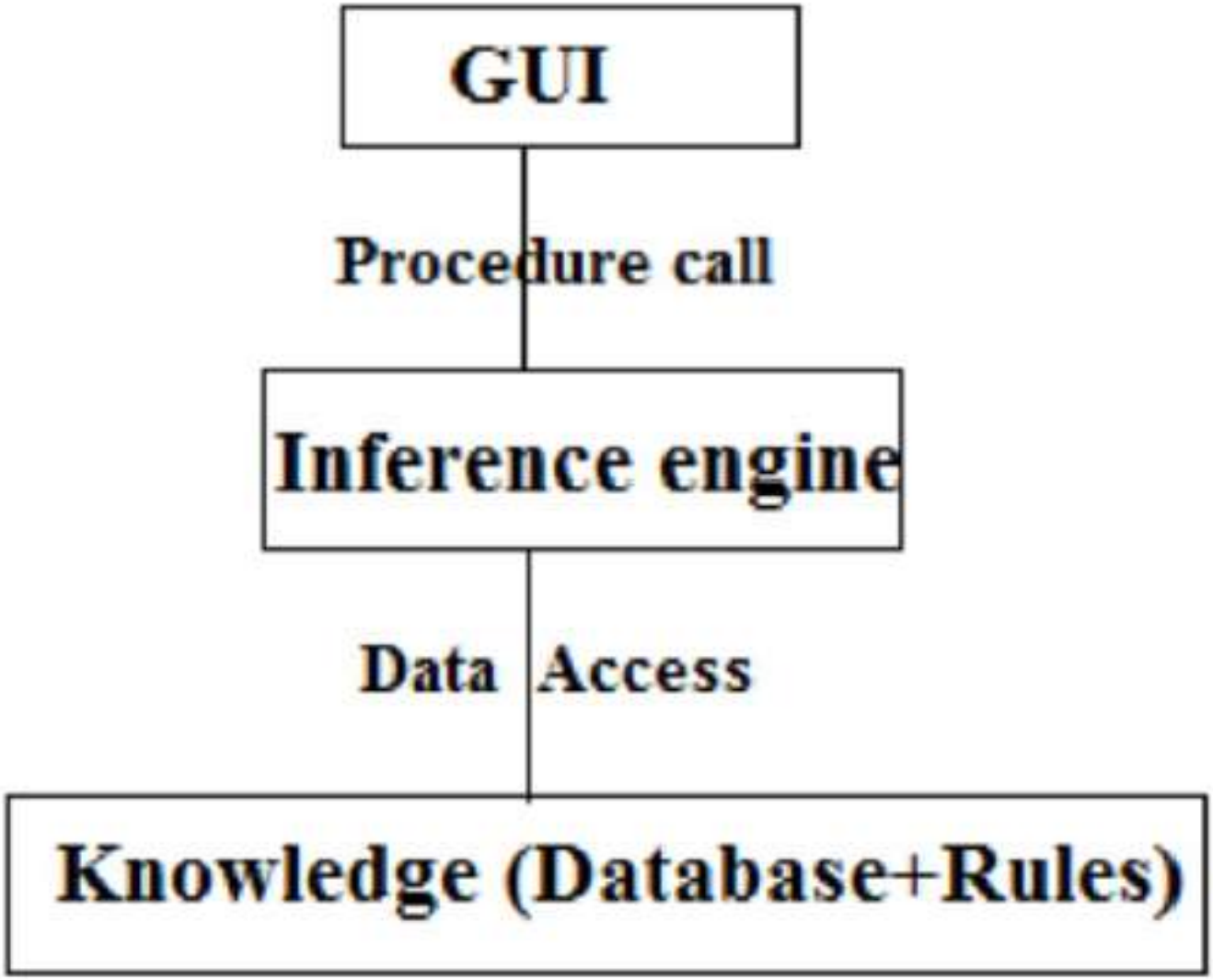


Fig: Rule-based style for Lunar Lander game

Interpreter style

- It executes the command dynamically one at a time, whereas the mobile code style involves the execution of one chunk of code at a time
- In this style, the commands are more general like Excel macros
- Many graphical editing programs are interpreter-based
- The interpreter interprets the program and issues drawing commands to the graphics engine
- Mobile code style transmits the code to a remote host machine for interpretation
- There can be various reasons for doing so:
  - Lack of computing power
  - Lack of resources (e.g. interpreter)
  - Large datasets that are remotely located
  - Mobile code can be of three different categories (depending on who requests the transmission, where it is transmitted and where the state is stored)

1. Code on demand has resource and state locally, but downloads code from remote host and executes locally
2. Remote evaluation is similar to grid computing
   - ✓ The initiator has the code, but lacks resources
   - ✓ so the code is sent to remote host for processing and the result is sent back to initiator
   - ✓ Mobile agent has code and state available on local machine, but lack some of the resources
   - ✓ hence code, state and some resources move to remote host where it is processed along with the remote host resource; and the processed data need not be sent back to the initiator
   - ✓ Use of this style is when processing is to be done on large datasets in distributed locations

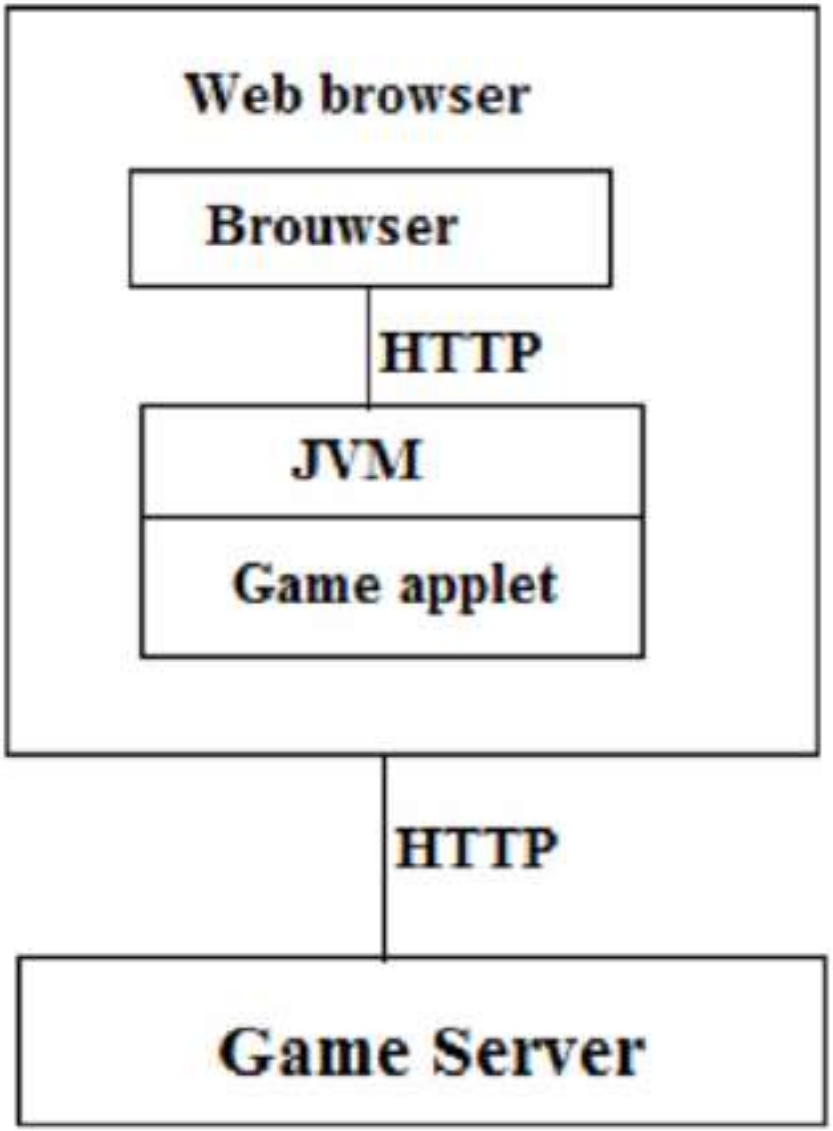


Fig: Mobile code style for Lunar Lander game

Implicit invocation

1. Push model
2. Pull model

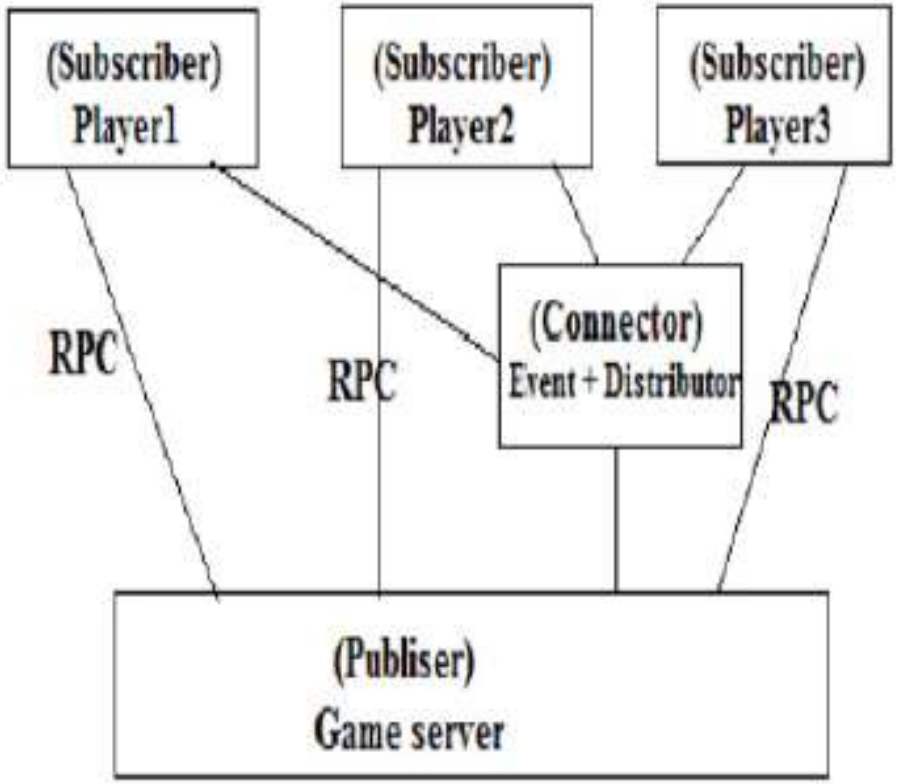


Fig: "Publish-Subscribe" style for Lunar Lander game

Peer-to-peer(P2P) style

- Peers are loosely connected, each acting both as a client and as a server

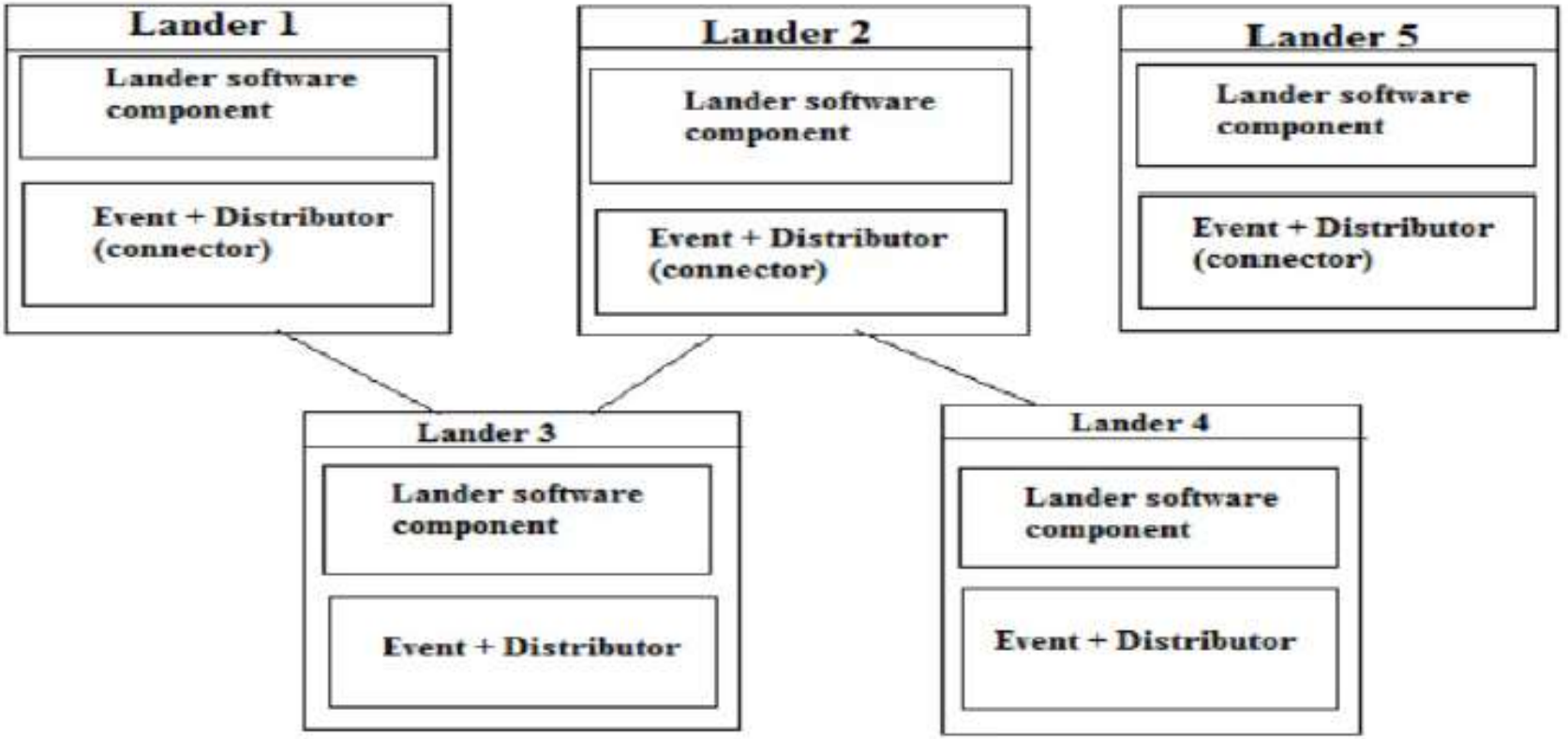


Fig: P2P style for Lunar Lander game

The tier system

- Software architects decides about what goes into each layer (tier) and logical verses physical separation
- Layers resides on the same machine, having same runtime (JVM) and any object in one layer can be passed to another by value or by reference
- The tier could be on different machines or on different runtimes, hence the data has to be passed by value only

- User interface tier code has to look up and call remote application tier methods using a middleware(e.g., .NET removing, web services, RMI, remote invocation of EJB, etc.).

Tier Zero

- Prototype is the tier zero of the development model
- Done at the beginning of defining the architecture in order to test our ideas of game design;
- It is also preproject investigation to efficiently implement tier-based architecture.
- Prototype types : Gameplay prototype, user interface prototype, subsystem prototype and algorithm prototype

Gameplay prototype is the most important one among all.

User interface (UI) prototype deals with user interaction details.

Subsystem prototype deals with subsystem interaction and the interfaces it need to export.

Algorithm prototype is abstracted behind an interface in order to “plug and play” the algorithm

Tier one and above

- The results of prototyping are used in tier one.
- The more game-specific functionality is implemented as soft architecture;
- The reusable and generic are implemented as hard architecture.
- Tier one begins at the lower level of granularity and moves towards the interface modules and the basic game framework code.
- The hard architecture (hardware-interface components and system menu components) is refined first and the later tiers mostly concentrate on adding more functionality through soft architecture.
- The game should be shippable (ready for release) at each tier, although all functionality may not be there, at least program should be free of bugs and the game should not crash.
- This will help restrict the additional until the current versions is working properly.

Architecture Design

- Architecture embodies is unifying concept and principle in its components and their relationship to each other and to the environment.
- A conceptual framework is needed to establish the terms and concepts and serve as a basic for evolution.
- Architectural description is a collection of documents, takes into account the stakeholder's viewpoint and concerns.
- Consistency among the architectural viewpoint has to be checked.
- Architectural environment is concern with developmental, programming and operational context of the system.

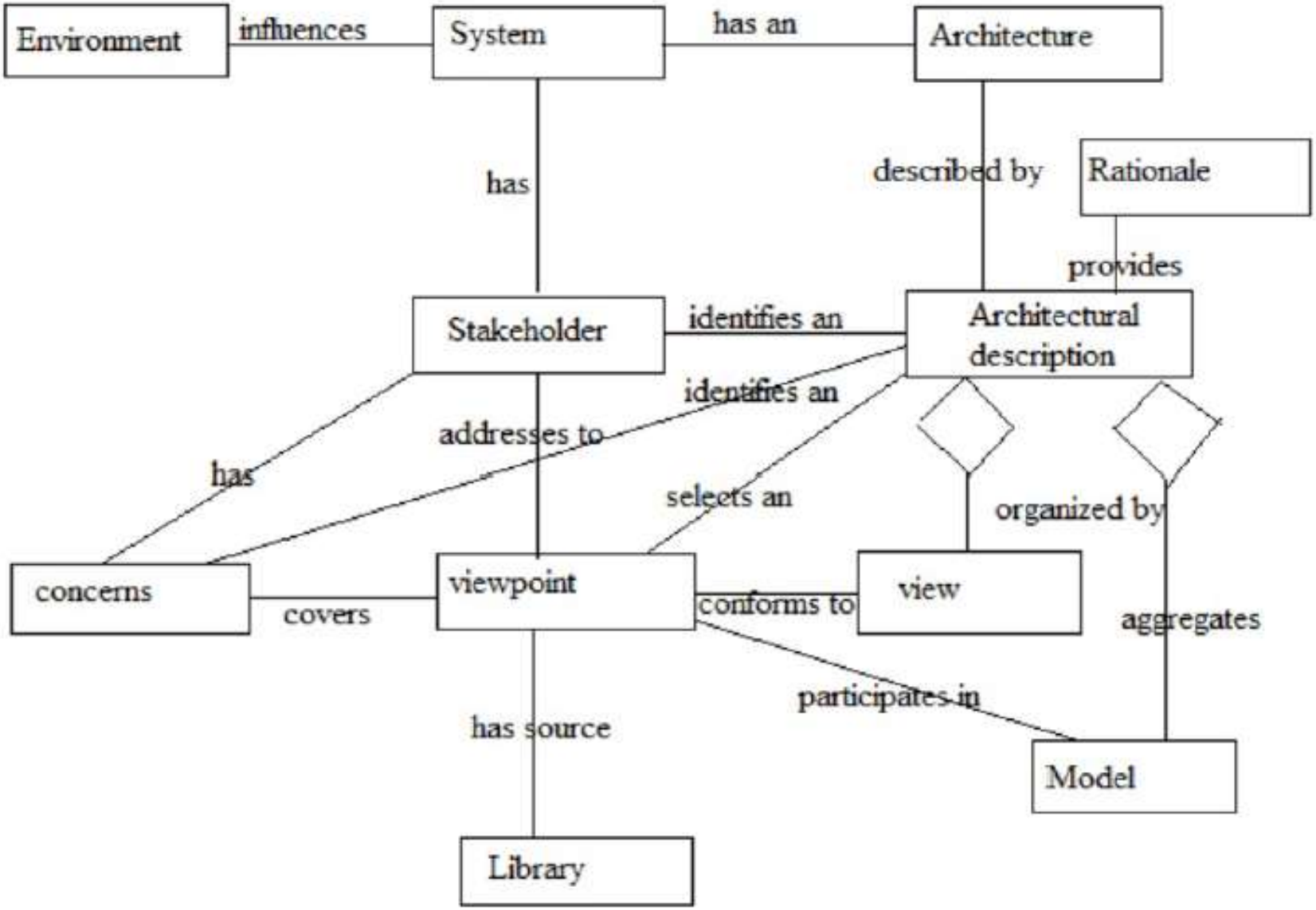


Fig: Conceptual framework

The structural viewpoint concerns:

- Identifying the elements of a system and of a system and its organization.
- Interface between those elements.
- The mechanism for their interconnection.

The architecture in game development stages: Prototyping, hard and soft architecture design.

- Prototype helps in tackling difficulties that might appear in the project.
- Hard architecture design lays the game framework-design of the system (system architecture – relationship between components)
- Soft architecture design incorporates the unique functionalities of the game and the data required such as graphics, sound, etc. - design of the game (game design architecture- relationship between tokens.

Applying tier-based approach to architecture design

At the basic level, game system has-

- User interface,
- event handler (bidirectional);
- data engine (level data, graphic data, etc);
- system behavior (collisions, general physics);
- game logic engine; game sound engine, game graphic engine;
- hardware abstraction layers (interface components);
- configuration system;
- menu system;
- music system;
- online help.

- Each of these subsystems can be used as a reusable component and can be reused across different projects. Most of these subsystems have standard interface so that each of them can be replaced with another if required with minimum of coding effort.
- When building a component-based architecture, there is a need to identify the existing reusable components first, both in-house and third-party components.
- For the first tier, the framework with minimum functionally for all the required components is built. The subsequent tiers fill the missing functionally in controlled stages.
- Interfaces form the most important aspect of technical design. Therefore, consistent interface between the subsystems is ensured.
- Each tier must all the requirements of higher level are taken into account.