

Production

FROM PROTOTYPE TO PRODUCT

Production

- After preproduction deliverables (eg. prototype or technology demonstrations) are accepted, you are free to proceed with production.
 - Development of the game based on the results from preproduction.
 - Testing of the game.
 - Release to manufacture.
 - Maintenance after release (typically in the form of patches and upgrades).

Development

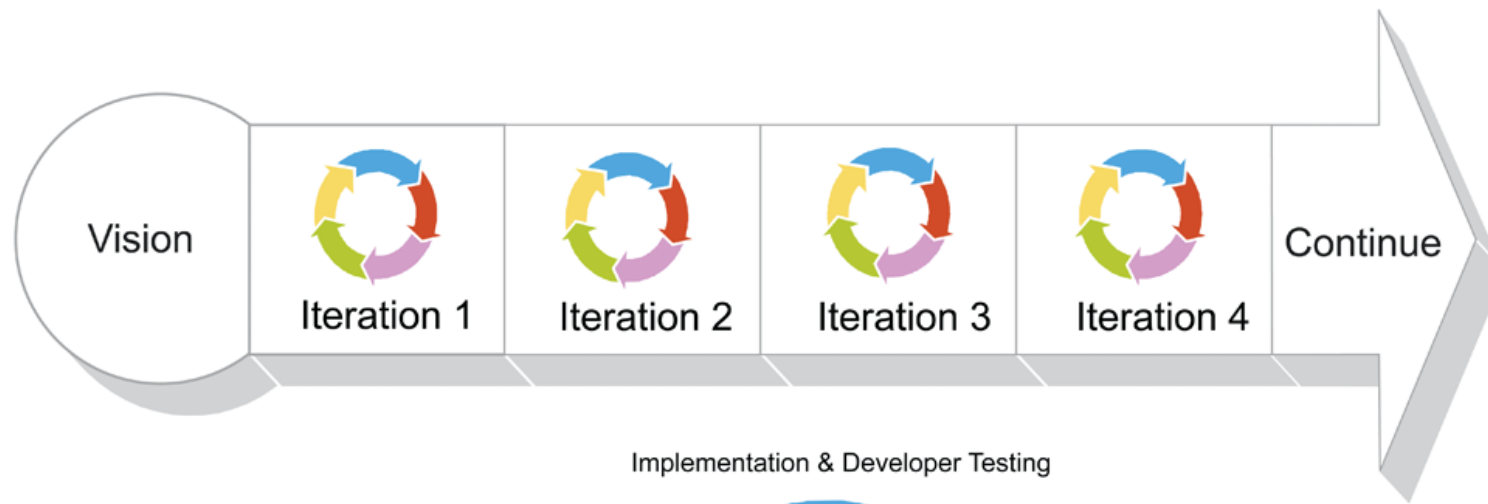
- Development is the long haul of video game production.
- Development of modern video games typically lasts six months to two years.
 - Very little can be done well in less than six months; there is simply too much to do.
 - Anything longer than two years, and you risk your game going stale or becoming obsolete before it is even released.

Development

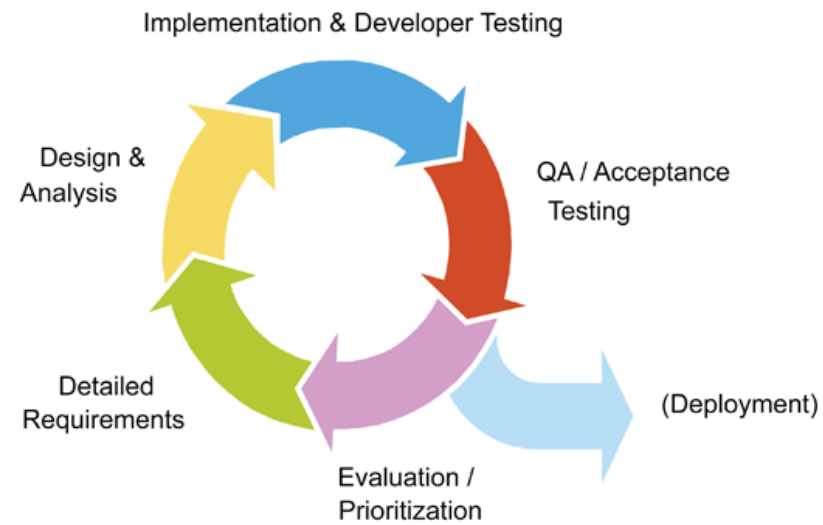
- Time is very deceptive at the start of development.
 - It would seem you have more than enough time to get everything done on schedule.
 - As deadlines near, panic sets in as you realize that you do not have as much time as you thought.
- It is critically important to break large tasks into small manageable tasks that can be rigorously tracked.
 - It is much easier to ensure that things are on time and progressing well this way.

Development

- Be prepared to revisit your designs throughout development.
 - As you discover what works, and what does not work so well, redesign will be needed and documentation will need updating.
- In the end, some form of iterative software development model might be necessary.
 - For example, some type of prototyping model (evolutionary or throwaway), or a spiral model (with risk analyses) might work best.



Iteration Detail



Scrum Iterations Cycles

Development

- There is a growing interest in the games industry in agile software development.
 - Agile methods differ from traditional iterative methods in that their time period is measured in weeks rather than months, and work is performed in a highly collaborative manner.
- Methods such as extreme programming, pair (or peer) programming, and the scrum development process might ultimately prove useful in the development of games.
 - Time and experience will tell what works best ...

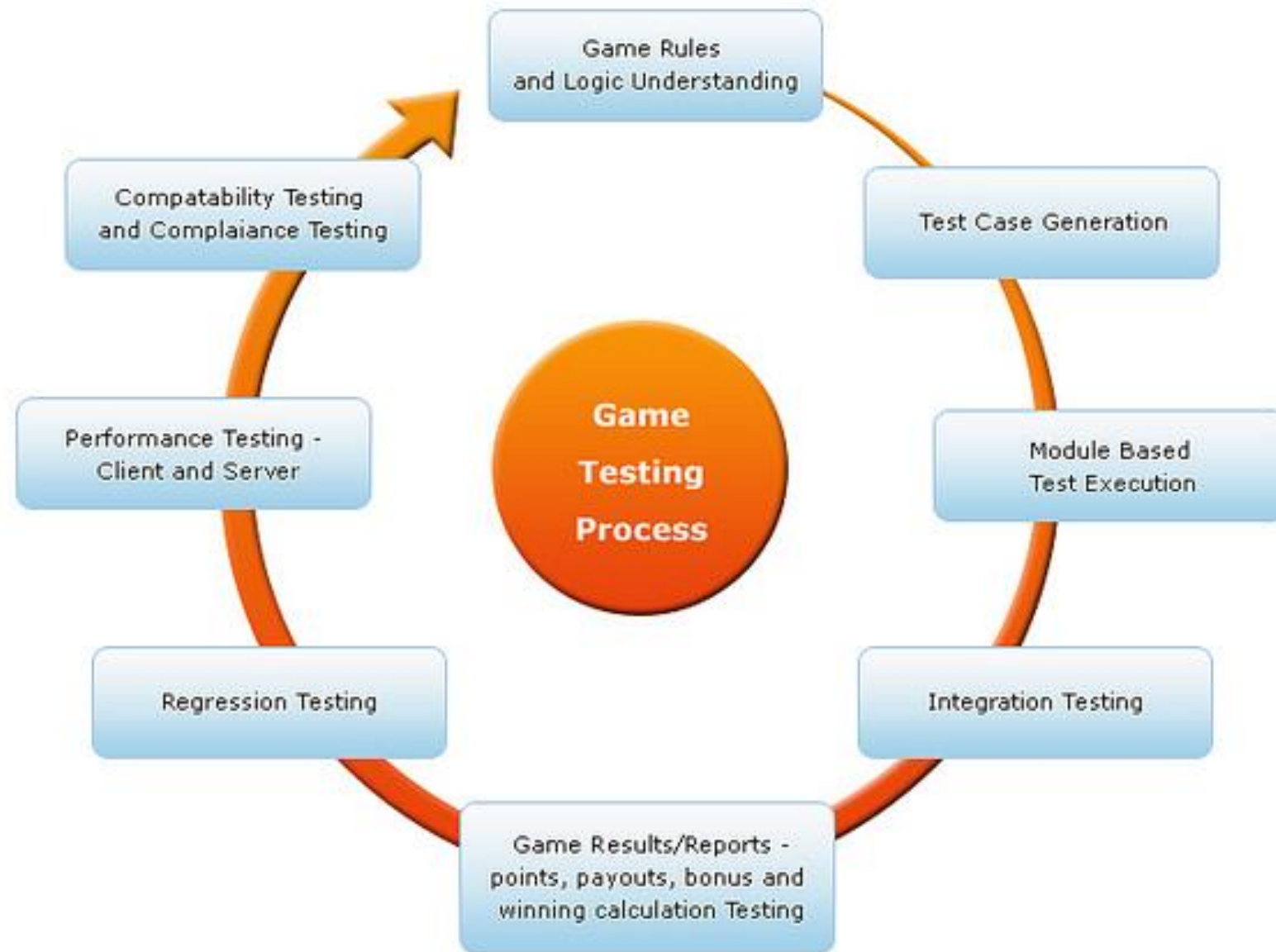
Development

- A few survival tips to keep in mind:
 - Maintain good communication across the development team.
 - Keep design documentation up to date.
 - Maintain the team's identity and spirit.
 - Give marketing and public relations the materials and demos they need – they will help keep people's spirits up when things get tough.
 - Be ready for a shock or two. When these happen, keep your head down and do the work! Things are rarely as bad as they seem.
 - Have a few features ready to throw away to help manage scope in the long run.

Testing

- Testing is important for both validation and verification purposes.
- Validation:
 - Are we building the right game?
 - To improve game design, gameplay, and so on.
- Verification:
 - Are we building the game right?
 - To eliminate bugs, remove imbalances, and so on.
- Testing should occur throughout development to remove problems as soon as possible.





Testing:

Different Types of Testing

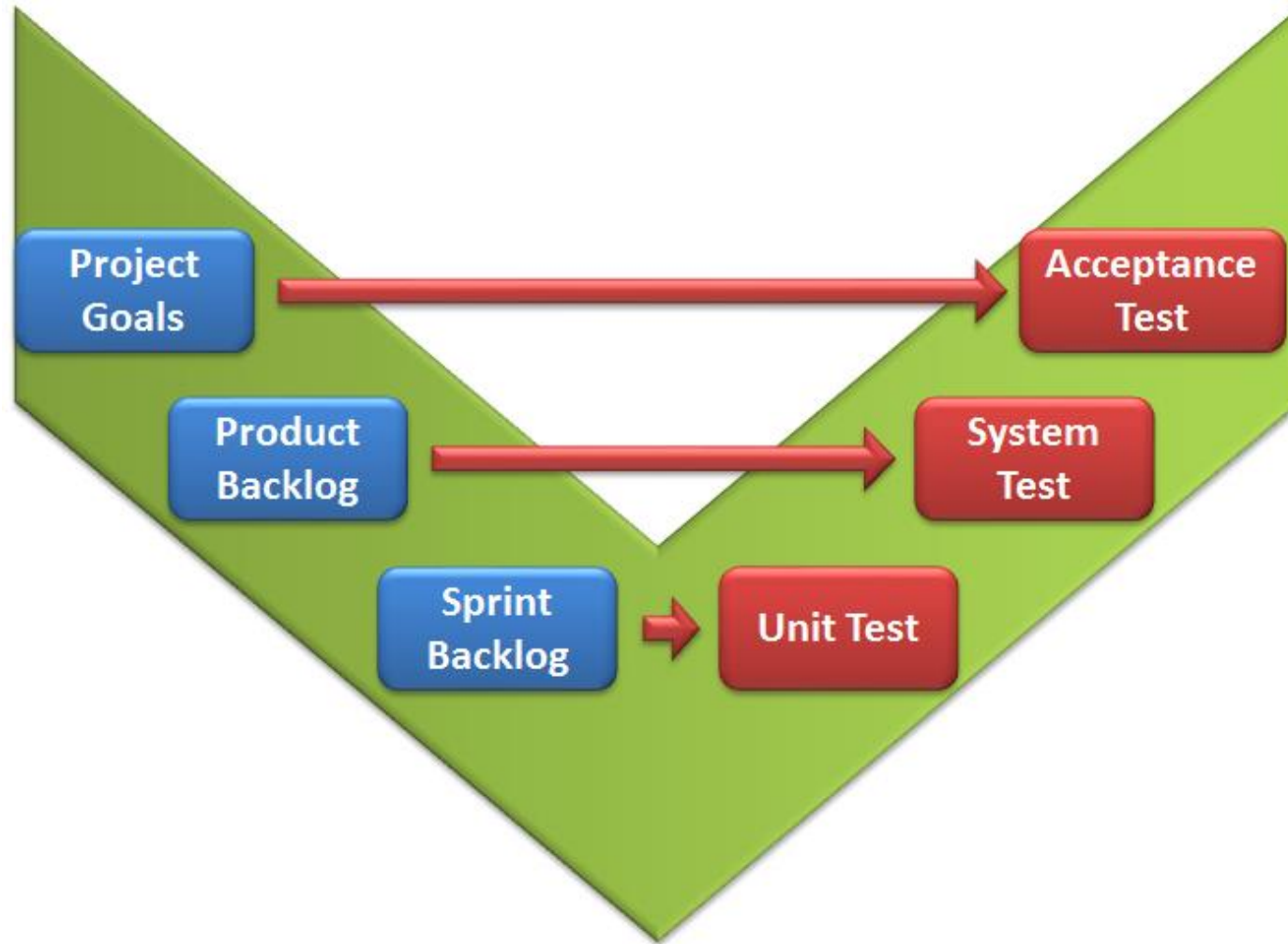
- Unit testing.
 - The testing of game modules on an individual basis.
- System testing.
 - The testing of integrated game modules as a more-or-less complete system.
- Acceptance testing.
 - An essentially complete game is demonstrated for acceptance and publishing.

Testing:

Approaches to Testing

- Black box (functional) testing:
 - Game functionality is tested according to specification, without looking at its internals.
- White box (structural) testing:
 - The game is tested according to its internal structure and code to ensure it behaves correctly when provided with test data.
- Regression testing:
 - Developing libraries of test cases that the game is sent through each time a change or update is made.
 - The purpose here is to retest the game to ensure it still works correctly after modifications.
 - Can be applied to both black box and white box testing equally well.

Test and Verification in Scrum



Code Freezes

- A code freeze can occur at many times during game development to prevent changes that could cause significant problems and delays.
 - Code freezes to fix key interfaces between modules to allow module developers to complete their modules without fear of modifications.
 - Code freezes to prevent new functionality or features from being added too late in the development process.
 - Code freezes in the last days of beta testing that allow only critical or “showstopper” bugs to be removed.
 - Code freezes before a milestone or deliverable.

Milestones

- Key milestones represent deliverables to the publisher.
- Often, there are several internal milestones as well.
- Key Milestones include:
 - First Playable (2nd, 3rd, 4th, etc may also exist)
 - Alpha, Beta, Gold

Milestones

■ Alpha

- Internal testing.
- The game is at the point where it is mostly playable from start to finish.
- Some content and gameplay might be missing, but the engine, interface, and other major subsystems are complete.
- The focus shifts from building to finishing; from creating to polishing.
- This is the beginning of the end!

Milestones

■ Beta

- Internal or external testing.
- Everything is now complete and integrated into an essentially finalized game.
- The goal here is to stabilize the game and eliminate remaining bugs before release.
- If possible, doing a public beta gets a lot of extra testing done for very little cost.
- The last portion of beta testing is crunch time, where the only important thing is finishing the game.

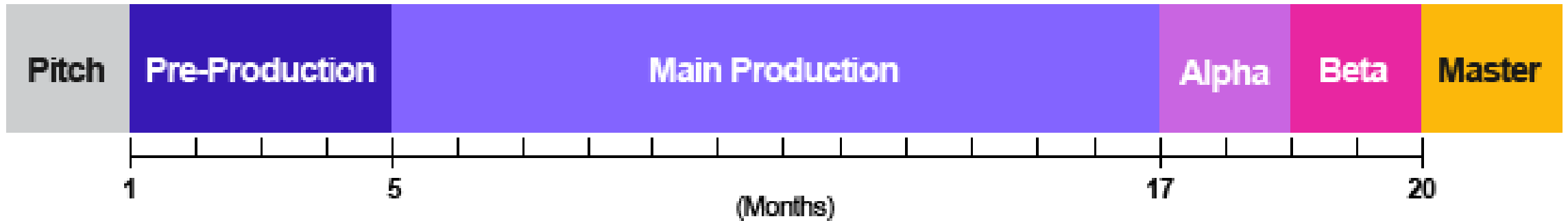
Milestones

- Gold candidate

- The game has been approved by the publisher.

- Gold master

- The game is released to manufacture when one of the gold candidate releases has been thoroughly tested and deemed acceptable by the console manufacturer.
- You can finally celebrate!!



Blitz Games Studios Milestones

Maintenance

- After release, the development is rarely over. There are often smaller releases that follow.
- Patches:
 - Typically to fix bugs discovered after release, or to handle incompatibilities with user hardware or software configurations.
- Upgrades and updates:
 - Represent additional content created to enhance the original game. Can be new levels, characters, weapons, story elements, and so on.
 - These are really mini-projects, and need to be handled as such, with the same management needs.

Pipelines in video game production

Introduction to pipelines

- The pipeline is a process where an asset or element of the game moves from concept to completion and then into the game build through a series of steps where multiple team members each contribute a portion to the overall asset.
- The notion of pipeline is not a new one! The car industry is using the pipeline for nearly a hundred years.
- In your games, always try to define, form and use pipelines.
 - That is because they will give you an structured way to produce things.

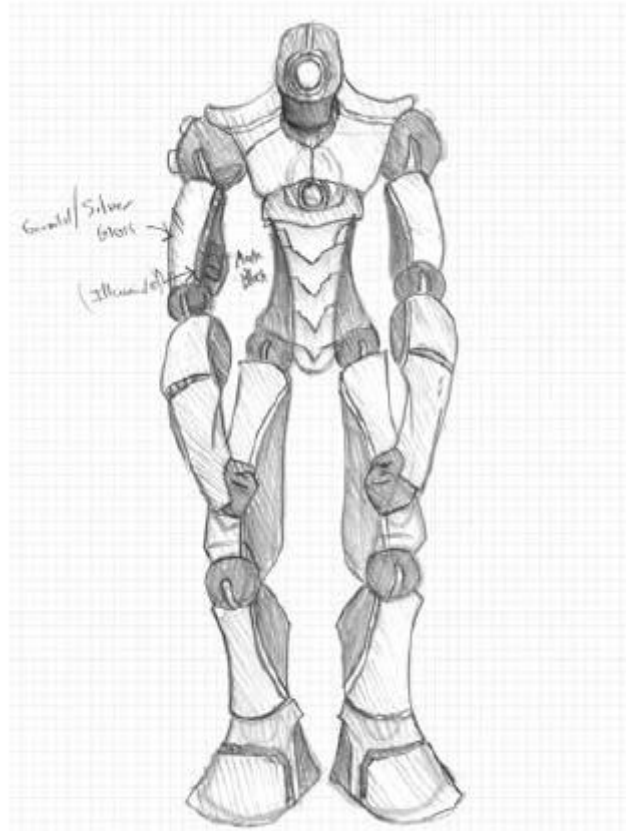
An example of a pipeline

- Game development consists of a series of interconnecting pipelines.
- Each discipline can have numerous sub-pipelines within it and as each element is completed it branches off onto another pipeline, eventually working its way into the final code.
- The clearest example of how a pipeline works is through the evolution of a character model.
- The example that follows is very basic; there are numerous techniques, styles, and approaches to a pipeline, this only being one.
- These pipelines are also never clear lines, having the asset back-and-forth at various stages, with work in progress moving onto the other pipelines so as to not cause delays in the other disciplines work.

Character modeling pipeline

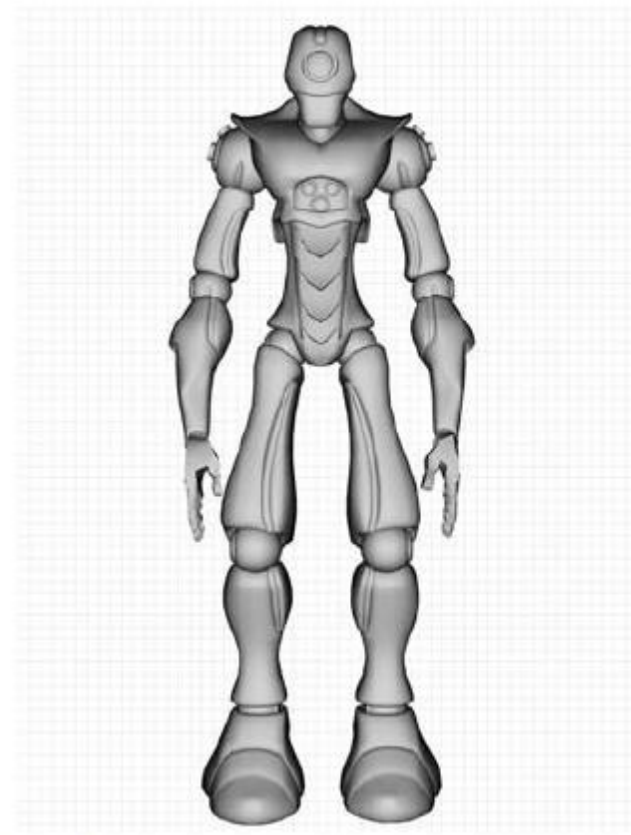
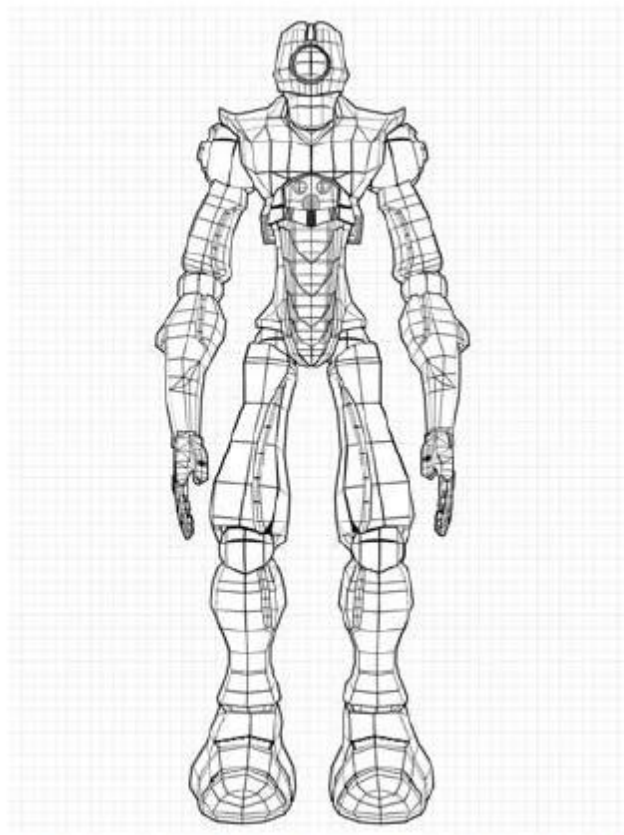
- A primary character model in a game is rarely made by one single artist from start to finish.
- Here is a simplified example showing the evolution of a character model as it goes through the artists ' pipeline

The modeling pipeline starts with a concept



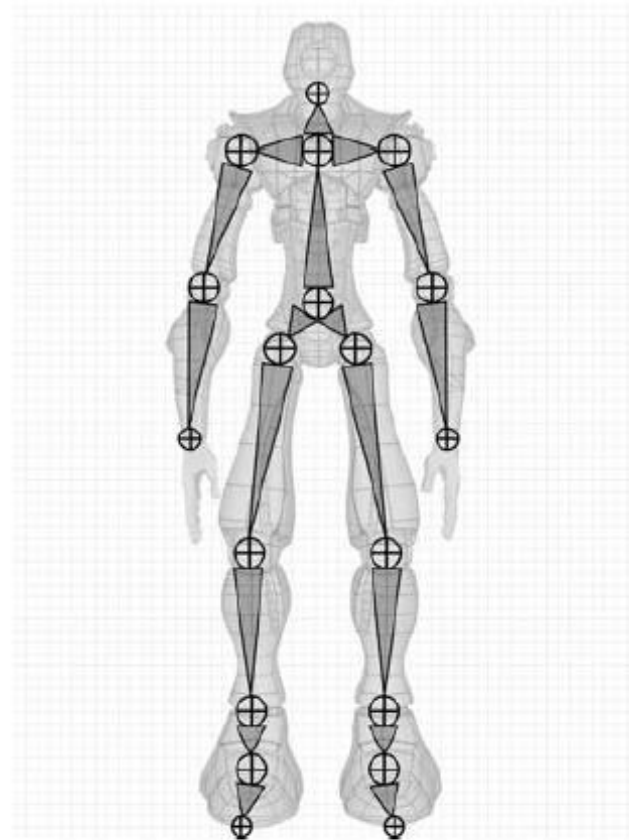
A concept artist creates a 2-D image of the character.

A 3d model is created from the concept



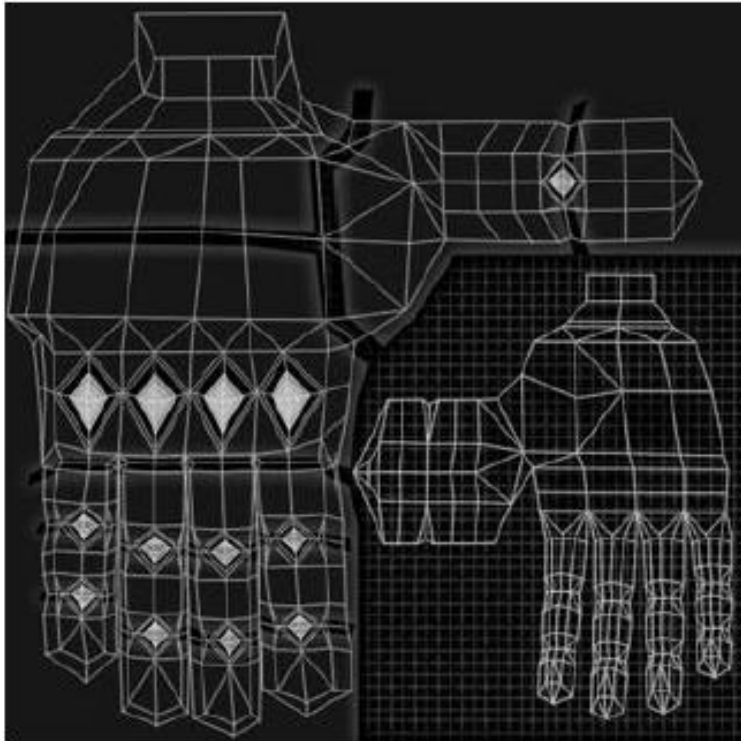
A 3-D model artist creates the 3-D model based on the concept.

Rigging artist makes the skeleton for the mesh

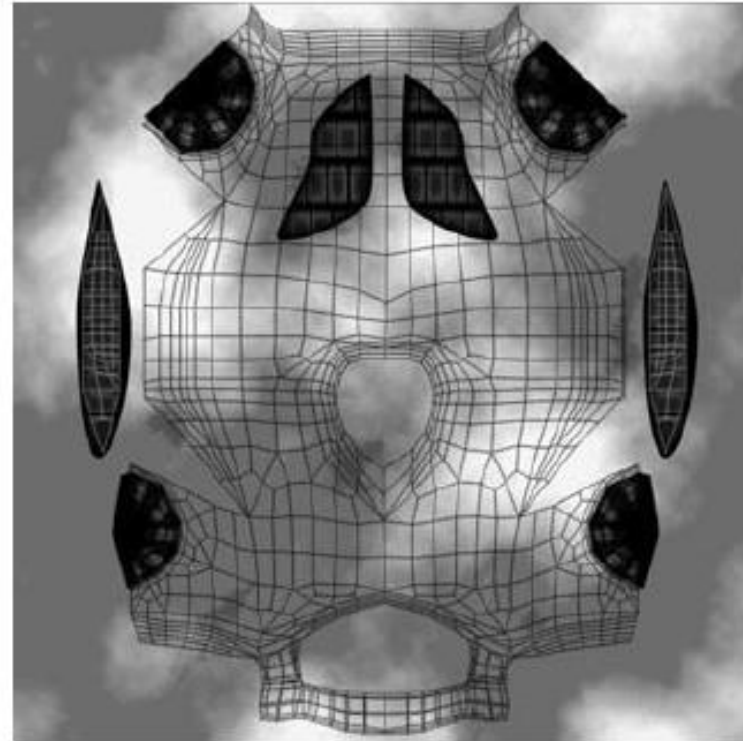


A rigging artist builds the skeleton (or bones) of the model.

A mapping artist creates the textures



A mapping artist then crafts the 2-D textures for the model (robot hand and upper chest textures).



A mapping artist maps the texture to 3D model



A mapping artist maps the texture to the 3-D model.

The final model is provided to animator who adds motion to the character



A final model is provided to an animator who adds motion to the character.

Now it's Programmers Turn

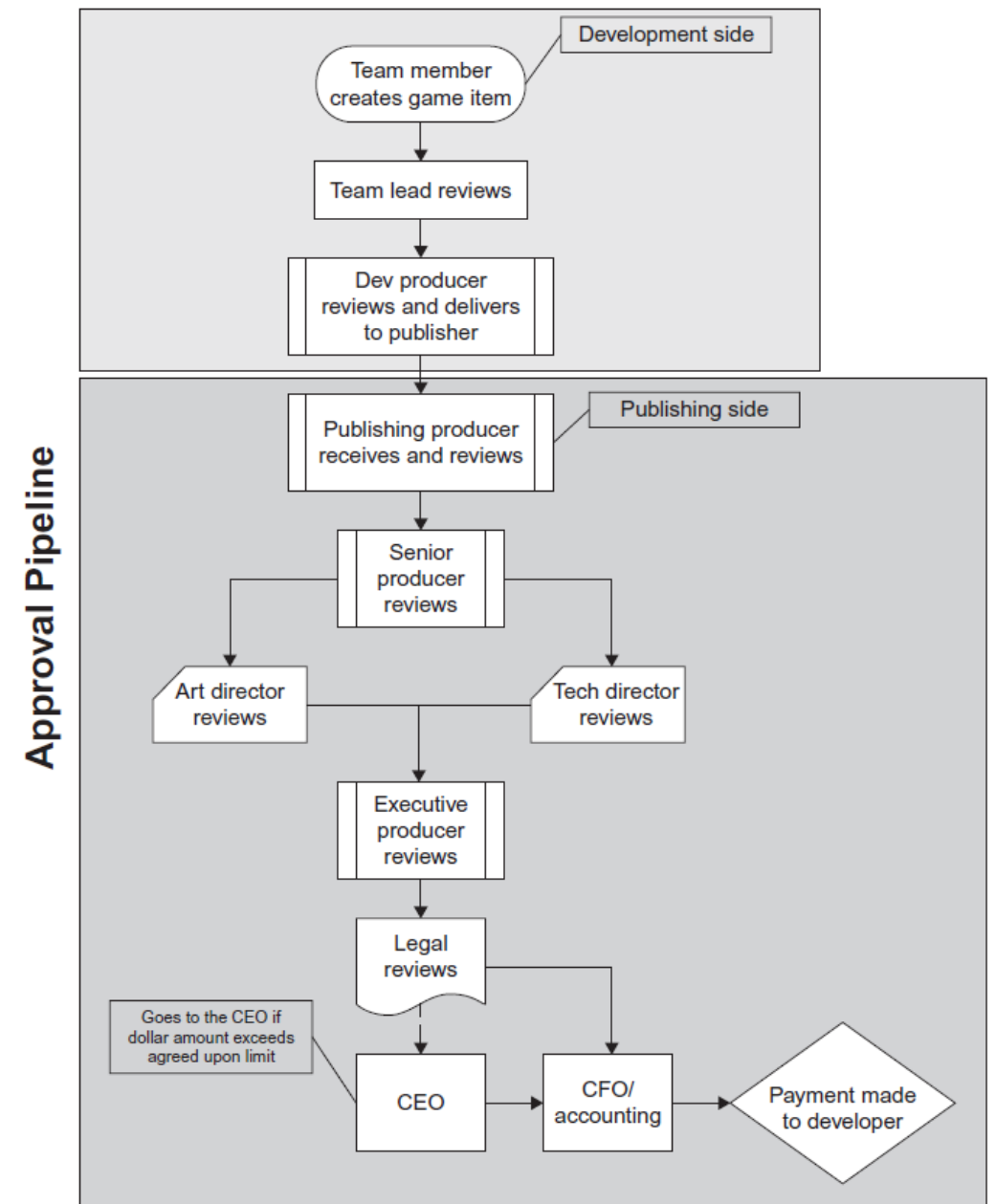
- Tools Programmers Creates a Way to Export Animated Model to the Game Engine
- Gameplay Programmers Sync Player movements with Game Logics.


```

1 void Update () {
2     if (!dead) {
3         float x = Input.GetAxis("Horizontal");
4         float absX = Mathf.Abs(x);
5
6         if (!hit) {
7             if (x > 0)
8                 skeletonAnimation.skeleton.FlipX = false;
9             else if (x < 0)
10                 skeletonAnimation.skeleton.FlipX = true;
11
12             if (absX > 0.7f) {
13                 SetAnimation(runAnimation, true);
14                 GetComponent<Rigidbody2D>().velocity = new Vector2(runVelocity * Mathf.Sign(x), GetComponent<Rigidbody2D>().velocity.y);
15             } else if (absX > 0) {
16                 SetAnimation(walkAnimation, true);
17                 GetComponent<Rigidbody2D>().velocity = new Vector2(walkVelocity * Mathf.Sign(x), GetComponent<Rigidbody2D>().velocity.y);
18             } else {
19                 SetAnimation(idleAnimation, true);
20                 GetComponent<Rigidbody2D>().velocity = new Vector2(0, GetComponent<Rigidbody2D>().velocity.y);
21             }
22         } else {
23             if (skeletonAnimation.state.GetCurrent(0).Animation.Name != hitAnimation)
24                 hit = false;
25         }
26     }
27 }

```

Even publishers use pipelines for the acceptance of a game



Example of a typical approval pipeline.

Post Mortem

- After your game is complete always use a post mortem phase. In this stage you will ask yourself two primary questions:
 - What went right?
 - What went wrong?