The Role of a QA Lead

Introduction

It takes a team to create a game. Someone to make the art, add sounds to the environment, design the GUI, plan the game play, write the code that ties it all together and makes it unique. Then the game must be played, to see if the art fits, the animation works, the sound plays, the code runs, and if it's all working... if the game is fun. Welcome to Quality Assurance, where anything that can go wrong is your job - to test, locate, describe, and decide when it's fixed.

How do I test a game that hasn't been made yet?

Just as there is design work to be done before a single line of code gets written, there is much to be tested and evaluated before anything resembling a game starts coming together. If there is work being done on the project, then there is work to be inspected, evaluated, and tested. Don't fall for the trap of being just a 'tester' – you are not. Your job is to insure that the game being produced meets the highest standards it can. Doing that means being involved every step of the way, from when the initial design is being created to the final steps of insuring the game is stable and running.

Good quality assurance is not easy.

Testing a game is a big job; responsibility for testing a game is even bigger. It requires a broad set of skills: the ability to organize people and data, the ability to constructively criticize, excellent communication skills, an understanding of all the disciplines within the game, the ability to analyze and solve problems with little or no data, and finally, the ability to learn and adapt to new challenges quickly. It is expected that QA Leads will join GAMBIT with little or no experience in their field – that's why we have QA group meetings, and a QA Mentor! Sara Verrilli is the QA Mentor, and her door is always open!

Things to think about when you think Quality Assurance:

- Understand the project, and understand the project's goals.
- Know what needs to be tested, and why.
- Think about who cares about this particular piece of testing, and what they want to know from it!
    - Is it for the user, so the game is easier to understand/ more playable/ more enjoyable?
    - Is it for the designer, to check how well this mechanic works, this level works, or this character works
    - Is it for an asset creator, to see how a character looks, sounds, moves, reacts? Or how well written a piece of text is?
    - Is it for the programmer, to confirm the game is running?
    - Is it for the client/product owner, to prove a research goal?
- If you are testing something no one cares about.... Why are you testing it?
- Conversely... At every point in the development cycle, you should be able to answer What's the most important thing to test right now? Why? For whom?
- Ask Questions! Ask your team what they are doing, what the intend to do, and how to do it. Ask Sara for help. Ask your Game Director what's most important right now.
- Ask more questions.
- Internal QA References you should read:
    - Focus Testing Basics
    - Usability Basics

- Milestones Timeline (for descriptions of milestone requirements)
- GAMBIT checklists (so you know what base standards your game has to pass at each milestones)
- Sample checklist set (so you have a checklist template, and an example of a checklist for a game.)
- Sample focus test documents.

So – what does a QA Lead do?

Responsibilities to the Development Team:

The QA Lead's job, first and foremost, is to **understand the current state of the game**, and to be able explain it to the development team, the product owner and the game director at all times. The development team knows which tasks they've completed; the producer knows where in the schedule the team is, according to what the team has done. But the QA Lead needs to know how all of those things are working - or not working - together. What is meant by 'state of the game'? Well, the QA Lead must know what in the game currently works (is playable), and what doesn't – and ideally, why.

As part of understanding the overall game state, the QA Lead needs to know what can be tested, what cannot be tested (and why!) and what needs testing. The QA Lead also needs to know *how* to test each the of the game's features, to confirm that the features work under all possible conditions, and across all possible player choices within the game. To help with that, the QA lead creates and maintains the ***test plan.***

The test plan details the expected testing needs for the game. Every game needs a different testing plan, depending on the technical and design challenges and innovations involved. However, almost all test plans need, at a minimum, the following:

- The target platform: what is the minimum CPU, RAM, and hard drive space you will take? For browser games, what OS/browser combinations do you intend to run on? How big will your download be (recommended: 10 MB or less!)
- The game's high risk features and components, and a testing plan to evaluate whether or not they are working. *(If you are making a multiplayer game, how are you (the lone QA tester) going to regularly test multiplayer functionality?)*
- A Features Checklist. A feature checklist is a list of standard user test cases used to test the complete functionality of the game. A complete feature checklist covers every aspect of gameplay, user interface, and every achievable game state from the user's point of view. A playthrough of a features checklist should mean that the tester has used and evaluated every functionality within the game.
- A Hardware/Software Checklist. A hardware & software checklist is a list of standard test configurations to test on, to insure the game runs on a variety of machines. For browser games, this may be a list of browsers and OS's. It should also
- Focus Testing plans – dates for focus tests, expected goals, and other focus testing documents.
- Usability testing plans – plan for usability evalutions, expected goals, evaluation results.

Obviously, as the scope and design for your game changes, so will the test plan. As some parts of the game become better understood, other areas may pose a higher risk. The questions the team needs to ask at each focus test will depend on choices made after previous tests. As more functionality comes into the game, the features checklist will need to be revised. The test plan is never finished – it is living document.

Responsibilities to the QA Team:

You are expected to attend all scheduled testing times (currently expected to be Tuesday and Wednesday, 3 – 5 PM) and to work with the other Lead Testers to schedule which games get tested during those times. You're responsible not only for testing your game, but also helping the other testers test theirs.

You are also responsible for working with the QA team to evaluate each Milestone Candidate against the GAMBIT standard checklist and the game's specific functional checklist, and to assist the other QA leads in doing hardware and software testing as needed.

"Testing." What does that mean?

The QA Lead is in charge of **all** game testing: play testing, technical testing, usability testing, and focus testing. Remember that all testing is inter-related, and at some level, overlapping. Play testing and technical testing, in particular, go hand in hand. Usability and focus testing are also natural partners.

Technical testing evaluates the game to make sure it is performing as expected. All features work *as designed*, all menu links work, the game is stable, it runs on all expected hardware and software, it meets legal requirements and studio standards. Technical testing also includes the subgenre of '**gamebreaking**.' What can you do to break the game? Use weird symbols in your name? Run off the edge of the map? Get stuck in random corners? Win the game using a stupid default strategy? Discover that one of the characters/powers/weapons is overpowered/underpowered? Click outside of the game area? Hit ESCAPE during the cutscenes? Press random combinations of keys to crash the game – and on, and on. If you can imagine a player doing it, they will – so you'd better test to see what happens to game by doing it first.

Play testing is the in house evaluation of design, fun, and playability of the game. If a feature isn't fun, or if the game is hard to use, or when a piece of art looks wrong or a sound is off.... that's play testing feedback, and it's a vital first test of the game's viability. Good playtesting data comes from not just the team's QA lead, but also from the rest of the team, and from the other QA leads. Offer to playtest another team's game, and get playtest data from their coders, artists, and other members. Anytime you can find a first time player – you should get their reaction and response!

Usability testing evaluates the overall usability of the game. While early usability testing can (and should be) done within the team and with experienced testers/UI designers, the final usability of your game can only be measured with new players – fresh users. How long does it take a **first time user** to go from starting up the game to playing it? How much instruction does a **first time player** need to play the game? Does your feedback help the player, and how well does the player understand his or her goals and abilities to achieve them in game?

Focus testing tests the playability, the fun, and the overall effectiveness of your game. Depending on your product owner's goals, it may also be the only way to determine if you are achieving your game's research goal. If you're trying to get a particular message across through your game – there is no way your team can confirm or deny your game's overall success. Once again, only **first time users** can give you the reactions you need to understand how your game is impacting and affecting its users. Focus testing is more than just finding 'first time users', however. Good Focus Testing also takes into account the game's intended audience, and ***focuses*** on gathering data from the type of players you expect to be playing your game. By observing the intended audience play, the QA Lead and the team can determine how actual users like or don't like the game and its specific features.

QA Tasks are Development Tasks

During a well managed sprint, at any time, the development tasks are clearly specified, and everyone knows what to do. Make sure that the QA tasks are equally well specified and accounted for. One way to do that is to create an accompanying 'test' task for each development task on the release cycle list. Another way to do it is to arrange for a 'test' column (if the team is using a task board.) As tasks are completed by the other developers, they are moved into the test column – and only the QA tester may move them to 'done.' Whatever method is chosen, QA tasks need to be accounted for.

A QA Motto to live by: It isn't DONE until it is done

On every project, every QA tester runs into the same problem again and again – so called 'untestable tasks'. Programmers in particular are prone to claim 'well, you can't see it, but it's there.' There is only one answer to that: an *untestable task* is not done. It is up person attempting to check in the task to enable the QA tester to test it *in the game*, if at all appropriate. Obviously, some deliverables don't go in game – design documents, or playable prototypes, for example – but those documents should still be read, and played, by the QA lead to confirm that they are completed.

QA Throughout the Development Cycle

As with all positions, exactly what a QA Lead is doing for the game depends on the stage of the game. Here's a rough breakdown of the phases of game development, and what the QA Lead should be focusing on.

Conception/Early Design (Reaching The First Design Review)

The very first goal is to come up with a game idea, then fill out to a full game design. Here is one of the places where the QA Lead needs to take off the 'critical' hat, and put on the 'creativity' one.

- Brainstorm! Throw out ideas, let them flow.
- Be creative! Think outside the box.
- Think of similar games, what you liked, what you didn't.
- Research! Are there any games on the web similar to what's being suggested? Is that good or bad? Any board, card, or other paper based games that might have similar mechanics, themes, or be otherwise related? Play them. Have your team play them.
- Imagine yourself playing the game as suggested.
- Think about game design ideas
    - Think about your intended audience.
    - What do you enjoy/what would they enjoy?
- Take ownership of the ideas being created, take part in the process of game creation. You need to like the game that your team is going to create, so you can be proud of your work.
- Prototype ideas instead of arguing, assuming, or guessing.
    - Don't argue about 'would this work or would that work' - make a prototype.
    - Don't say "I think", "I guess", "I wonder" - test it out and see.
- Make it fast, make it ugly, make it work. Twenty minutes spent making and playing an idea is worth hours of talk.
    - Especially in conception/early design, it's fine to test 'just' with the team. If you want to go outside of the team, keep the testing in small groups - you can analyze and use data from one person at a time, but in a group, you start missing stuff.

- Keep prototypes small - it's ok to test just one feature, not the whole game idea. Think about what NEEDS to be tested, and WHY. WHY are you testing this feature? WHAT do you want to learn about it?
- If you're taking a prototype outside the team, make sure you UNDERSTAND how to run it, and WHAT you are trying to test. What is the key game mechanic? Why have you chosen it? How do you expect players to react?
- Prototyping is playtesting/focus testing – do it right, so you aren't wasting your time. Gather data, and use it.
- Review and think about the Design Review requirements. Is your team on track to meet them? What are they missing? What else do you need to do?

Preproduction: From Design to First Playable

During pre-production, the team is deciding how to make their game, how long it will take to code features and generate assets. The QA Lead needs to create a test plan, generate and give feedback on the design ideas, and understand everything about the project plan that he or she can. Understand why the team is making or discarding the design choices; make sure that the team is making decisions based on data, not just "I thinks." Get time estimates from your team members; check in on them when they're out of time.

- Keep prototyping! Don't be afraid to change ideas, if there's a good reason to.
    - Go ahead and test outside the team with your prototypes, but keep them small and fast. Look for patterns in the way people use your prototype, both 'good' and 'bad' feedback. Take notes!
    - Share your results with the team!
    - Ask them what they are worried about! Can you test it? How?
- Communicate!
    - Work with the designer to decide how game elements work.
        - Keep researching - do you have design problems? Interface problems? Are there games out there that have solved it? What mistakes did they make that your team can avoid?
    - Peek over the Artist's shoulders. (Don't pester!)
        - Understand what kind of an aesthetic they are aiming at!
    - Work with the producer - for every task the team creates, make sure there's a way to test it!
        - Don't let 'untestable' tasks make it into production. *Every task should have a test condition attached to it by the time it's on the task list for the milestone!*
    - Read everybody else's Role Document, if you haven't already.
- Get technical!
    - Learn how to build the game, learn how to use FogBugz, master Perforce.
    - Understand the asset pipelines.
        - Know how to put a piece of art or a sound in the game! Know where they go!
        - Go ahead and **do** it, with a scratch piece of art and a placeholder sound.
    - Learn how to use the debugger.
    - Are you a programmer? Can you assist the programming team by writing unit tests, or assisting with writing code to assist in testing?
- Keep up with the documents!
    - Keep your team honest! Make sure that if it gets decided, it gets recorded somewhere - in a game design doc, in a task list, in an asset list.
- Make a Test Plan!
    - Technical stuff:

        - What's going to be hardest to test about this project?
        - What kind of checklists are you going to need?
            - Make a skeleton checklist: what features does your game have to have for first runnable? First playable?
        - What legal and studio standards does the team have to meet?
        - How much time is it going to take to do a full play through of this game? (ie, to play the game such that you've hit every feature, asset, win and/or loss condition in game)
    - Playtesting stuff:
        - Who's your intended audience? How are you going to get feedback from them? (Are they likely to come to the two evening Open Houses? How can you get them to?)
        - Think about focus testing - start recording questions the team generates that can only be answered with users.
        - What does the Product Owner want out of this game? How can you test for that?
- Test the First Runnable against the First Runnable Requirements. Does it pass?

PRODUCTION (UP THROUGH BETA)

Production is when the game gets made. The development team is turning out code and assets, and assembling it into a playable game. Technical testing and user testing go hand in hand here - technical testing to make sure everything works, and user testing to make sure the game stays on goal as features get added in. The sooner your game is testable, the better your game will be.

- Fight to get the game testable!
    - Encourage your team to get functionality in first. The more that is working, the more that can be tested.
        - 'Ugly' placeholder art is easier to replace than bad mechanics!
        - 'Answer the research question' FIRST
        - ''find the gameplay!'
        - Prototyping may still be a good way to 'first run' tasks that are taking too long to get in, but don't slow the team down with it.
            - Don't test just within your team! Get fresh faces to look at your prototypes and your early game!
            - Know what the important questions to answer are, whether you are playtesting, user testing, or focus testing!
- **PLAN** your testing, and make sure it meshes with the team's needs.
- Make sure the builds run!
    - At the end of the day, every day, the build should work.
    - Don't allow the game to be 'unplayable' for any amount of time. You have the responsibility to inform the team when the build is down, and they have the responsibility to fix it and keep it up.
        - Lend a hand! If the build is 'suddenly' broken, track down which change broke the build, and find out, as best you can, what broke the build. That will save time in fixing the build.
    - Make and test your own builds.
        - Learn what the errors mean, so you can better pinpoint what broke the build.
- Make sure tasks that say they are 'done' really are 'done'
    - Do a bit of technical testing every day.

        - Start hardware/software testing early – start checking browsers, play on someone else's machine, look at how big your game's footprint is. Does it lag on a netbook?
        - Stay on top of the task list - test things as soon as they go in.
        - Keep the bug count at zero!
            - Confirm fixes promptly - get things Resolved and Closed!
            - Understand why a bug isn't getting fixed.
        - Keep an eye on the buglist - if bugs are hanging out for more than a day, find out why. Should it be killed? Is it a task to put off till a different sprint?
        - Remember everyone on the team is busy!
- Use the group testing time!
    - Different eyes see different problems - spend time testing other projects, and get others to test yours.
    - Use this for both technical and play testing feedback!
    - Don't be shy about getting fresh eyes from within the studio to give you an hour of testing.
    - It's not too early to start looking at compatibility!
        - Does your game run on all browsers?
        - All common OS's (Mac, Windows XP, Vista?
        - Does it run with common stuff open in the background (chats, music players, office software?
        - What happens if you open/close/surf away from your browser?
- Be a testing czar!
    - Make sure your team is playing the game a bit every day.
    - Update your test plan and checklists!
    - Make sure the Feature Checklist is current, and shows
        - Which features are currently in and working
        - Which features are intended to be completed for this sprint
        - What the core, required mechanics are (and, if they aren't implemented yet, when the team expects them to be)
        - How you are fulfilling the GAMBIT standards requirements.
    - Use your feature checklist!
        - Testing the game against the feature list allows for regression testing - it makes sure that everything that was working still is.
    - Keep checking compatibility!
- Speak up - politely!
    - ASK QUESTIONS.
        - What does the team need to know about the game?
        - What does the Pwner need to know about the game?
        - How, and what, can you test to find THOSE answers?
    - Give the team your opinion about how things are working.
    - Collect and organize feedback from other players and testers, and present it to the team as clearly as possible.
    - Make suggestions for improving features, changing designs.
        - A great game needs testers who care about the game design, and who make it fun!
    - Understand that not all of your suggestions are possible.
    - DON'T CRITICIZE PEOPLE.
        - Be critical of the game, but remember, real people are putting real work into it! Never be rude in a bug report!

        - Be clear about what is an opinion and what is a fact.
        - ALWAYS keep the research goal in mind!
- Keep an eye out for the milestone deliverables!
    - Review the required milestone checklists, and make sure that anything required for them is on the task list!
    - The day before a deliverable, at least, do a run through using the checklists. Does your game pass?
- Stay on top of the documentation!
    - Is it being updated? Is it accurate? Is it useful?
    - If written info is required, but no one has time to do it - do you?
    - "If it ain't useful, find something that is."
- Organize focus testing!
    - Find out what your team wants out of focus testing!
    - Find out what your product owner wants out of focus testing!
        - These may or may not be the same thing!
    - Make sure you'll have actual intended audience members to test!
    - Make sure your team knows how to conduct themselves during the test.
    - What data do you need?
    - How are you going to get it?
    - How are you going to review it and use it?
    - Observer scripts (for your team to follow as they sit folks down to play the game.) + observer notes (to remind your team members what they should be looking for/taking notes on.
    - Hand outs for players if your UI is still incomplete/not working.
    - Surveys for players to fill out afterwards – keep them short and to the point. Only ask questions you want the answers to!
- Think about usability!
    - You're using it the most - remember what seems easy or natural to you may not be for most people!
    - Watch people play the game for the first time. Does everyone have trouble with the same thing?
    - Talk to Marleigh (our Interaction Design Director) if you don't like the feedback and don't know what to do about it!
    - Get help with usability issues – earlier is always better.

POLISH (FROM BETA TO RELEASE CANDIDATE 0)

Here is the last chance to make the game great. Take it.

- Test the game. Test, test, test, and then test some more.
    - Have other people play the game!
    - Have the development team play the game!
- Take focus test feedback seriously!
    - Understand the problems, think seriously about the fixes.
    - Understand why you think the fix will work.
    - Put it in, and test it again!
- Usability! If it doesn't work well now, it never will!
    - It's worth taking the time to fix things that don't 'feel' right, or 'play' right!
    - Is it too hard? Probably.

        - Remember that the 'average' tester is going to play the game for half an hour, not forty five hours!
        - Does your game appeal to the player on the first screen, in the first ten seconds, with the first sound?
        - If you can't attract the player in the first fifteen seconds, when will you?
- Standards! Check credits, copyrights, splash screens, legal requirements!
- Make sure it's always running.
- Keep bug count at zero! There's no time left for dangling tasks!
- Update your feature lists! You should know what the final game is going to look like, and your checklists need to reflect that.
- Do regression testing! Don't let any new bugs slip in!
    - Don't let your team be surprised by feature failure at the last minute!
- Once again, review the checklists before the Release Candidate.
    - *Don't be surprised if you fail to pass your first release candidate - but don't be surprised at WHY you fail!*

GOING GOLD

This is it- the game your players will play.

- Make sure your checklists are up to date.
- Offer to check someone else's game.
- Remember, you can't Certify your own game as Gold - one of the other QA testers has to do that.
- Think stable, think standards, think SHIP