GAMES AS A SERVICE
Lessons learned from 4+ years of Awesomenauts on Steam

Ronimo Games
Robin Meijer & Joost van Dongen
Welcome!

- Robin Meijer, producer
- Joost van Dongen, lead-programmer and co-founder
- Ronimo Games, indie studio making games since 2007
Introduction

- Awesomenauts
- 3v3 Online action-platforming
- Launched in 2012
- Seven years of development so far
- Games as a service business model
Introduction

• Talk about the Games as a Service model
• Why we use this model
• Lessons we learned while supporting Awesomenauts since 2012
• Walk you through an update cycle, how we actually update our game
Games as a Service

After releasing a game you can...

<table>
<thead>
<tr>
<th>MAKE A SEQUEL</th>
<th>MAKE ANOTHER GAME</th>
<th>GAME AS A SERVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Take lessons from your game / reception</td>
<td>Take lessons from development</td>
<td>Learn about your game from players</td>
</tr>
<tr>
<td>Make another, better / bigger one</td>
<td>Make an unrelated game</td>
<td>Grow existing project in relevant ways</td>
</tr>
<tr>
<td>Focus on what made your game do well</td>
<td>Focus on what you can do well</td>
<td>Focus on truly finishing what you started</td>
</tr>
</tbody>
</table>

- Take lessons from your game / reception
- Make another, better / bigger one
- Focus on what made your game do well

- Take lessons from development
- Make an unrelated game
- Focus on what you can do well

- Learn about your game from players
- Grow existing project in relevant ways
- Focus on truly finishing what you started
Games as a Service

After releasing a game you can…

<table>
<thead>
<tr>
<th>MAKE A SEQUEL</th>
<th>MAKE ANOTHER GAME</th>
<th>GAME AS A SERVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Take lessons from your game / reception Make another, better / bigger one Focus on what made your game do well</td>
<td>Take lessons from development Make an unrelated game Focus on what you can do well</td>
<td>Learn about your game from players Grow existing project in relevant ways Focus on truly finishing what you started</td>
</tr>
</tbody>
</table>

For Games as a Service, you need:

- Ability for players to remain engaged for long periods of time (endless replayability)
- Some way to generate revenue from engaged players
- Some way to expand on existing content in meaningful way
Our vision on Games as a Service

• Putting player-count above immediate revenue.
• Keeping the game fresh by releasing updates, giving players a reason to come back.
Why...

• Player-developer interaction
• Realizing our true vision
• Fairly easy on Steam, even as an indie
• Working on a live game is amazing
• Actually lots of fun
What does that mean?

- Around 69% of Steam revenue past Year 1
- Share of DLC vs. game sales in revenue increases
- Additional revenue outside Steam, enabled by ease of doing this on Steam.
- Over 2 million copies across platforms
LESSONS LEARNED FROM FOUR YEARS OF AWESOMENAUTS
August 2012

- Launch Awesomenauts
- Future content essential part of our product description.
- Wanted to deliver on that soon, to show commitment
August 2012

- Launch Awesomenauts
- Future content essential part of our product description.
- Wanted to deliver on that soon, to show commitment
- Content was already done.
September 2012

- Released first cosmetic DLC
- Revenue from existing players
- Share of revenue grows over the years
- Players talk about ‘supporting us’
- Need to experiment with content / price to see what resonates with players

Distribution of Revenue since August 2014
June 2013

• Getting better at communicating with players
• Forum topic with hints about new character
• Vague hints got community engaged
• Hardly any work
• Important lesson about communicating with our most hardcore fans.
August 2013

- Releasing content when it’s done
- Updates not bringing players back
- Growing number of ‘dormant’ players
- External forces result in players (Steam Sales, bundles, content creators)
- Players didn’t have any idea of our long-term plans
- Kickstarter campaign to realize ‘next level’ Awesomenauts
August 2013

• Pitching Awesomenauts: Starstorm

• Massive success

• Players respond well to long-term vision as narrative resonated with what they wanted

• Players wanted to help us make this happen

• Post-launch crowdfunding is possible
• Don’t make promises. They’ll haunt you.
• Talk as much as you can about your vision
• Be open about things you can talk about
March 2014

- Successful launch on Playstation 4
- Ports capitalize further on existing content
- Total revenue much smaller than Steam, still easily worth it
- Games As A Service difficult on console
April 2014

- Dev Days 2014:
  Robin Walker, Communication and Community in Games as Service
- Bundling updates into one, address problem of ineffective updates
- Moving updates beyond patch notes
April 2014

- Named updates.
- Teasing content.
- Engaging community, getting them involved weeks ahead of launch.
April 2014

- Updates resulted in player spikes
- Returning players
- Reaching players who didn’t play anymore
April 2014

- Reaching out to 'dormant' players
- Giving them a reason to come back
- Presenting narrative, not just patch notes
- Reaching out to them through Steam events, announcements.
- Providing incentives for 'following' you.
December 2015

- Continued doing updates this way.
- Impact slowly decreasing.
- Facing same issue again.
- Bundling updates further, narratives.
December 2015

- Meta-update
- Sharing content more than 6 months ahead
- Press picks up the story again
- Plenty to look forward to
- Players assured of future of the game
- New DLC drives revenue
December 2015

- Players return for launch of the new mega-update
- Lots of new content to experience
- Refreshes game, after four years
December 2015

- Selling gameplay content as DLC generates revenue
- Poor communication
- Player backlash
- Balancing free vs. paid content
2017

- Sticking with the narratives around updates, metapatches to communicate vision.
- Big stuff coming for Awesomenauts
- Still delivering on some old promises
- Not announcing anything yet
A PATCH: FROM CONCEPTION TO HOTFIX
Awesomenauts patching process

- Develop and iterate internally
- Several weeks: open betas
- Thursday: make build for QA
- Friday: QA company
- Monday-Wednesday: fixes and new build
- Wednesday: launch
- Friday: hotfix
- Few weeks later: balance patch
- Months later: console patch
Prerequisite: patching on Steam

- Steam makes patches super easy
- Patch can go live immediately
- Can do multiple patches per day
- Can revert to previous patch
- Can launch new DLC yourself (if prepared)
Choosing what to make

Sources:
- Our own ideas
- Community requests
- Issues in the live game
- Long term vision
- Things previously cancelled
Player feedback

- Separate emotion from fact
- Players get angry and always disagree
- Players dislike stagnation, want new things AND hate change
- Dev must grow a thick skin
Balance

- Games As A Service is awesome for balancing
- Can do lots of balance patches
- Players expect demand constant balance tweaks
- Need player feedback and metrics to do balance
- Perfect balance does not exist
Balance and the meta

- Must regularly change even if balance already 'perfect'
- Over time players flock towards same tactics
- Boring: everyone does same thing
- Even if tactic hardly overpowered
- Players also discover new tactics over time
Our current balance method

- Designer reads forums
- Discusses in conference calls and Twitch streams
- Makes list of planned changes
- Lets players give feedback on list
- Implements changes
- Beta
- Change based on feedback
Internal development

• Prototyping!
• Brainstorming!
• Playtesting!
• Iterating!
• Not talking about this today!
• Muhaha!
Using Public Beta instead of QA

- Each patch needs testing
- Extensive QA *every month* too expensive for small dev
- Also need player feedback
- Solution: replace most QA with betas
Ways to distribute betas

• Outside Steam
• Pre-launch on main app
• Separate Steam app
• Switchable Steam branch
• Dual-loaded DLC
• Live beta for everyone
Betas: Separate Steam app

- Completely separate application
- Users can enable/disable download
- Doesn’t share economy/workshop/achievements
- Need to set up and maintain Steamworks twice
- Need help from Valve to set this up
Betas: Switchable Steam branch

- Standard Steam feature
- User selects beta
- Can be behind password
- Hackers can see beta even without password
- User needs to download patch with every switch
- Easiest way to do betas
**Betas: Dual-loaded DLC**

- Put beta in separate DLC depot
- User enables free DLC to download and keep updated
- Ask on startup which version to run
- No download needed when user switches
- Entire game on disk twice
- Best way if repeated switching discourages users
- Valve likes this because they don’t need to do anything by hand for us
Betas: Live beta for everyone

- For doing temporary betas on everyone
- Entire game on disk twice
- We check our own server to see which to run
- Quick switch
- Large playerbase
- Beta with everyone = bugs for everyone
Giving access to betas

- Open beta for everyone
- ‘Secret’ password
- As a Kickstarter reward
- Paid beta access
Getting players into betas

- Only first beta of new content has many players
- Matchmaking problematic with few beta players
- Stale feedback after first round
- Do marketing, play beta on Twitch
Community translations

- Fans willing to help translate
- Build a small network of translators
- Let them check each other’s work
- Can be super fast
- Reward: Golden Duck icon
- Use professionals when too complex / big
Internal version management

- Lots of content in various stages of development
- Release one thing without getting bugs from another
- Standard solution: branches

We dislike branches:
- Too many merge conflicts when we refactor
- New content ‘hidden’ internally
Internal version management

Our solution:

- Most development in trunk
- Branch off release build 1 week before patch
- Disable unfinished content in release branch
- Hotfixes based on this branch
Hiding unreleased content

- Players hack game to find hidden content
- Must remove content entirely or accept leaks
- Our solution: tool automatically removes assets from builds
Paid QA

- Paid QA with a specialized company
- We only do this for big releases
- Limited testing of new content to reduce price
- Send build to QA through Steam beta branch (behind password)
- QA always on Friday, decide on Monday before
Final fixes

- Monday to Wednesday
- Fix issues found in QA
- Limited testing internally (~4 people 2 hours)
Why release on Wednesday?

- Not weekend: want to be around on release
- Not Thursday/Friday: weekend work if hotfix needed
- Not Tuesday: Steam server maintenance
- Not Monday: day before Steam server maintenance
Releasing the patch

- We do everything ourselves
- Except store items pre-approved by Valve
- Release patch on Wednesday evening (European time)
- Valve Seattle awake in case of emergencies
Patching problems on Steam

- Often broken files
- Can take up to 24 hours before user receives update
- Can require restarting Steam
- Won’t download while game is open
- Period with different versions
Hotfix

- Every patch has issues
- Despite internal testing, QA and betas
- Solving this too expensive
- Our solution: hotfix on Friday
- Result: users dislike bugs, but appreciate fast hotfixes
Balance fix

• Gameplay content gets balance fix after several weeks

Why?

• Beta not played enough to find all exploits
• Takes time to find best tactics
• Feedback first few days ‘incorrect’
Patching and beta on console

- Certification makes patches super slow
- Patch without new DLC: 1 week minimum, 2 weeks realistic
- Patch with new DLC: 2 weeks minimum, 4 weeks realistic
- No easy way to do betas
Games as a service on console

- Better than last generation, still really difficult
- Turnaround time too long
- Must plan much more
- Can’t fix quickly when needed
- Requires more QA
Our solution for console

- Release everything on Steam first
- Wait for balance to settle on Steam
- Create console patch afterwards
- Result: PS4/X1 always months behind
- Console players hate this
CONCLUSION
Conclusion

• Developing for a live game is super fun
• Games As A Service can provide continuous income
• Steam is awesome for Games As A Service
• Must plan first patches before launch
• Group content into big patches for marketing
• Uses betas extensively
• Learn when and how to communicate with the community
• Develop a thick skin
Want to talk?

• Joost van Dongen
  joost@ronimo-games.com
  Dev blog: www.joostvandongen.com

• Robin Meijer
  robin@ronimo-games.com