Example Document:

Reward Mechanics Primer

Building Worlds To Teach Science

Concepts for Designing Optimal Reward Systems

Abstract

Optimal reward systems make a person's investment of time both meaningful and fun.

Table of Contents

Reward Integration – Habitats

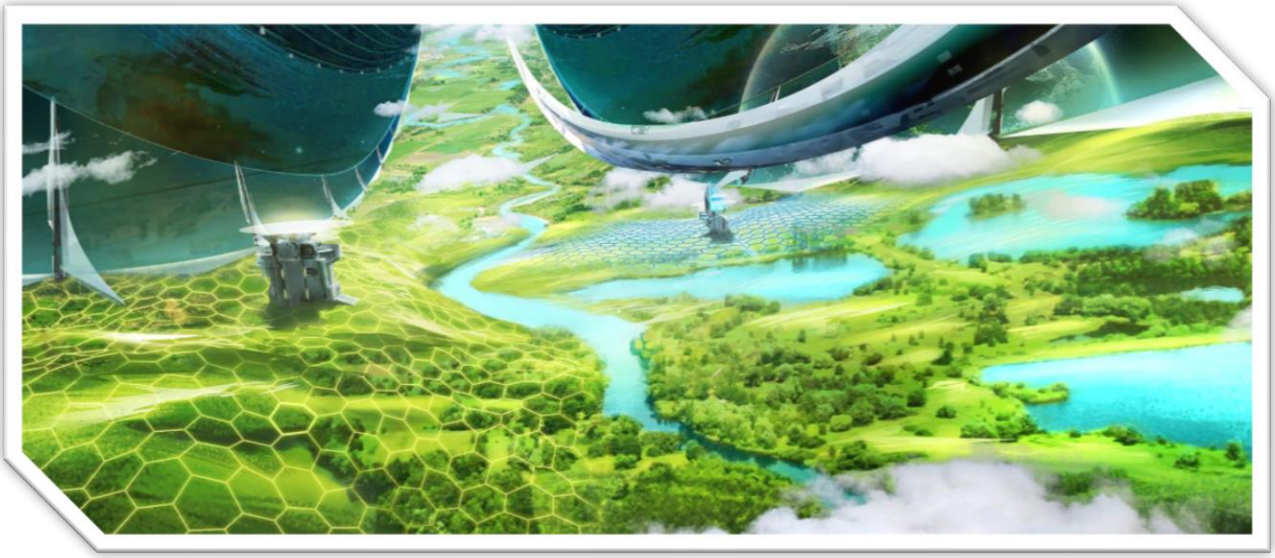
At a basic level, the reward system needs to support desired outcomes in student performance. These outcomes may be quantitative or qualitative, yet the reward system must be integrated to support both.

Another consideration is making the rewards system fun and engaging, allowing students to choose their learning (when allowed by the teacher or proctor) in such a way that their progress can be scripted by the teacher or self-directed.

The reward system contemplated for the product here is a 'building' game, similar to SimCity or Civilization. Students build an environment to support exotic forms of life. These environments are largely self-supporting, and rather than using fail states (negative reinforcement or 'black hat gamification') we will choose instead to reward optimization (positive reinforcement or 'white hat gamification'). This positive reinforcement rewards players that spend more time tuning their environment but does not penalize students who leave their environment largely untended.

Basic System

The environment building system will use a contiguous land/sea area. The edges will be closed off artificially, either by using the conceit of the space station's walls or energy fields. The following is an early concept drawing of how the environment could look.

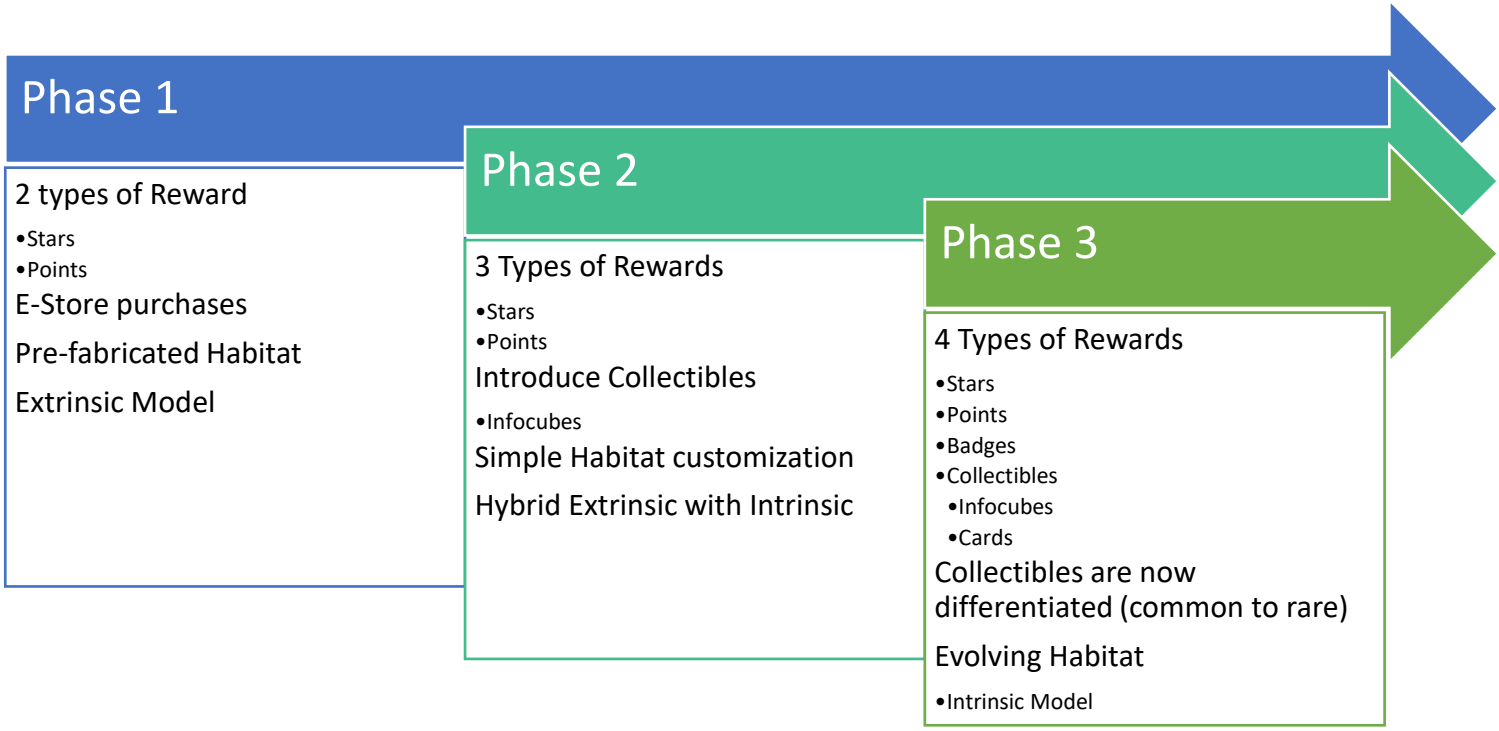


The hex grid is an early concept of how we delineate areas the student may affect using a combination of upgrades, power-ups (items or powers that let you increase the effectiveness of your habitat pieces or the habitat itself), and items to customize their landscape.

Phased Development

Our development efforts will start with a minimally viable product and limited feature set. From there, we will expand the habitat system via phases where increasingly more complex concepts and features can be tested for engagement and fun.

At the simplest level, our system will follow a reward loop that earns students stars or points for successfully completing missions, scoring on assessments, collecting objects in the environment, and completing mission or learning objectives. In this context, a 'mission' includes all aspects of the Topic's instructional model.



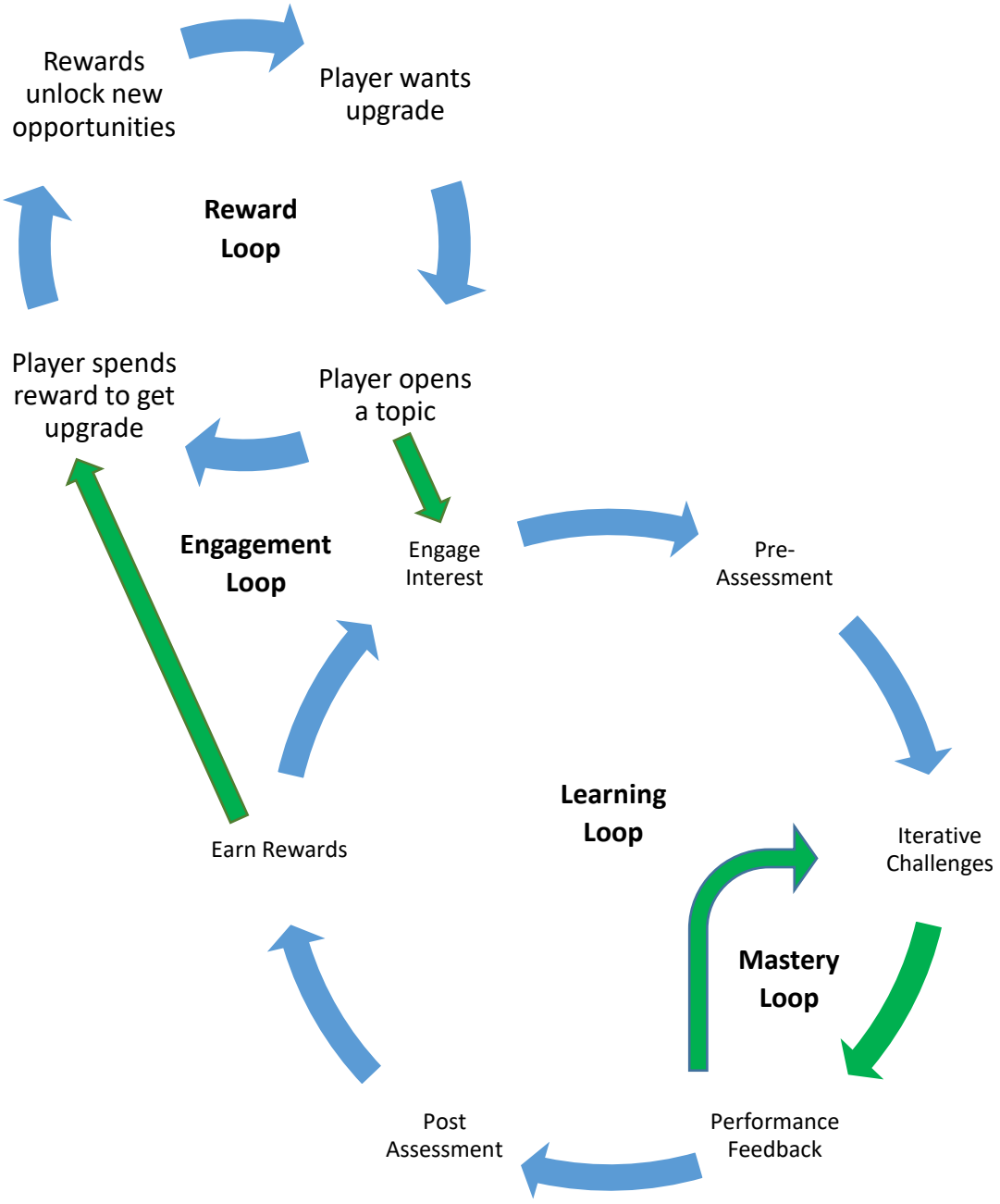
These points will be used to buy objects and upgrades for the habitat. As students wish to customize their habitat's look, they will need to earn more points, thereby completing the loop.

Note the following graph shows how players are motivated. While it is not imperative to use every method displayed here, using different techniques to complement and support a reward mechanic is vital to player engagement and motivation.

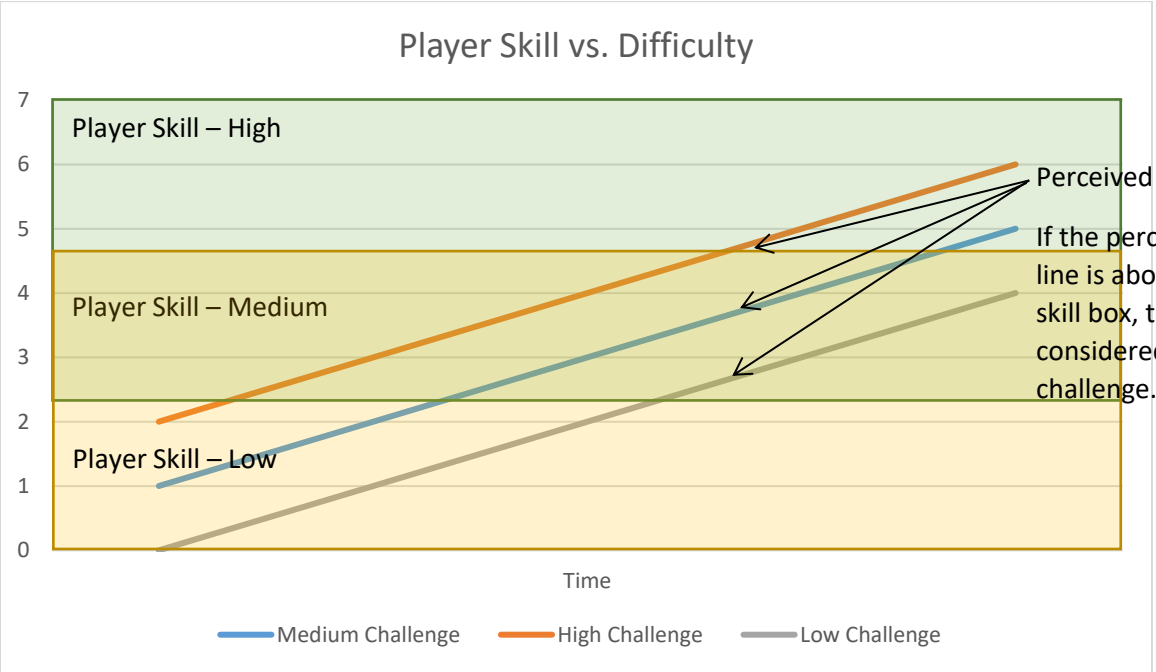
Chou, Y. (2014, June 14). Octalysis: Complete Gamification Framework - Yu-kai Chou. Retrieved January 11, 2016, from http://www.yukaichou.com/gamification- examples/octalysis-complete-gamification-framework/

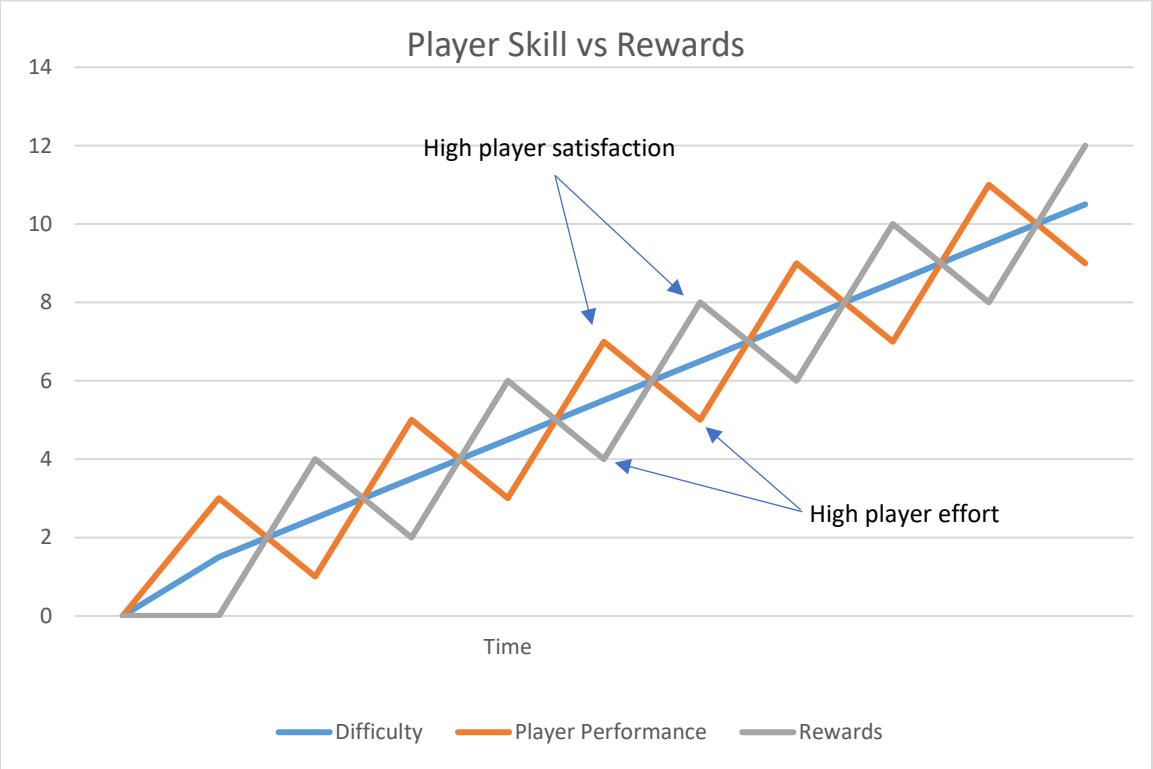
Multiple Loops

Multiple loops (rewards, engagement, learning, and mastery) are used to synergistically support each other.



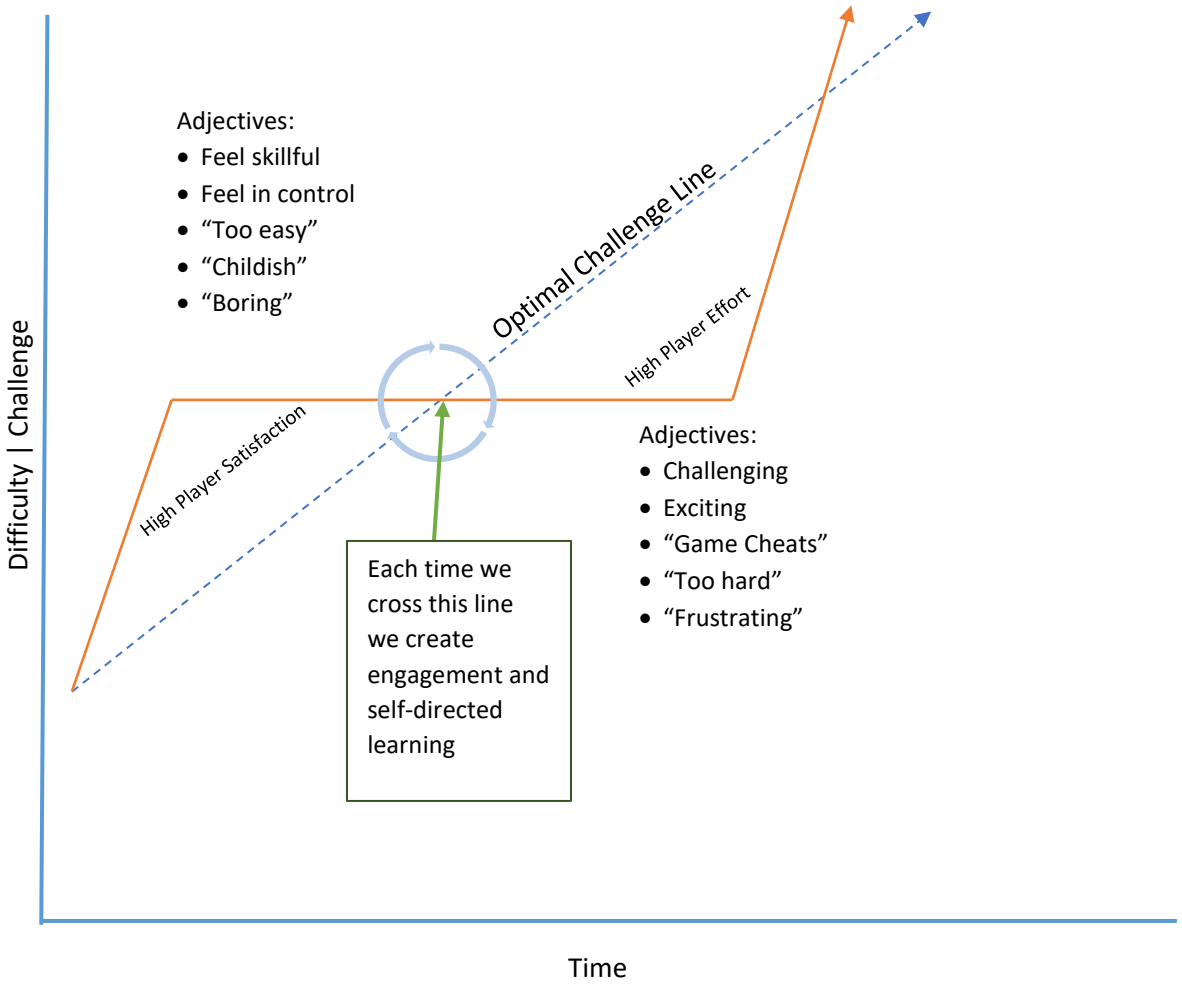
GAMEPLAY – REWARDS AND CHALLENGES





Note that as the player moves above the difficulty line (performs better) the reward incentive drops below the difficulty line. This is the 'satisfaction' curve. The player enjoys playing because he's better than the game. As the player drops below the difficulty line (performs poorer) the reward system scales up, providing incentive to improve. This is the 'challenge' curve, promoting effort to succeed.

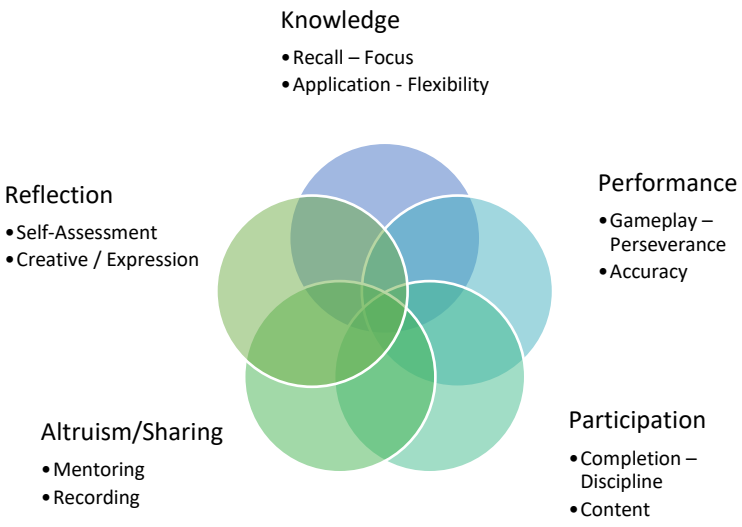
ADAPTIVE DIFFICULTY



TYPES OF REWARDS

What Activities Do We Want to Incentivize?

1. Knowledge
    a. Recall – Focus
    b. Application - Flexibility
2. Performance
    a. Gameplay – Perseverance
    b. Accuracy
3. Participation
    a. Completion – Discipline
    b. Content
4. Altruism/Sharing
    a. Mentoring
    b. Recording
5. Reflection
    a. Self-Assessment
    b. Creative / Expression



To properly motivate a student, the reward system must incent each of these activities. Therefore, four basic rewards may be earned:

Points (currency)
- Basic currency used to purchase items and upgrades from the 'Store' for your Habitat. We will create the proper fiction to justify this (nanoreplication, etc). Points are awarded primarily for Quantitative performance (Ex: scoring on a test)

Stars (replayability
- Designed to reward completing gameplay, boosts replayability. Potentially be used to buy special items from the Store that may only be available at certain times. Used to denote players who have exceeded performance expectations through finishing a level completely. (Ex: Angry Birds)

Collectibles (low clickthru)
- Designed to reward interacting with all our rich media content, reducing 'click thru'. (Ex: finding all infocubes and watching them) Can be either found by scanning a creature or plant, or be awarded for special circumstances designed into the Challenge. Collectibles does not need to be bought and can be seeded into a players habitat for free. It is a special reward for players that uncover information via infocubes, video, and other info sources.

Badges (cheaper upgrades)
- Designed to make improving your Habitat cheaper. Students earn badges by completing Topics, much like they earn Stars by completing gameplay. Each level of Badge earned would reduce the costs of certain items, collectibles and powerups for the Habitat. For example: A level 2 Photosynthesis Badge would cut the cost of Prairie Grass by 50%.

Why not just one type of reward?

Players love earning different types of rewards for different types of play. As a feeling of self-fulfillment, different types of rewards provide players with multiple chances to be recognized for excellence. Note

the diagram previously showing the different types of motivations for the player. We are utilizing only four extrinsic rewards, and one very powerful intrinsic reward, the Habitat's ability to evolve and customize itself to the player's wishes.

To support our theory that multiple rewards are correlated to increased performance, here is the abstract from Dr. Christian R. Shelton of MIT:

> *Christian R. Shelton Artificial Intelligence Lab Massachusetts Institute of Technology Cambridge, MA 02139 cshelton@ai.mit.edu*
>
> ***Abstract***
>
> *For many problems which would be natural for reinforcement learning**, the reward signal is not a single scalar value but has multiple scalar components**. Examples of such problems include agents with multiple goals and agents with multiple users. Creating a single reward value by combining the multiple components can throw away vital information and can lead to incorrect solutions. We describe the multiple reward source problem and discuss the problems with applying traditional reinforcement learning. We then present an new algorithm for finding a solution and results on simulated environments.*
>
> ***Conclusions***
>
> *It is difficult to conceive of a method for providing a single reward signal that would result in the solution shown and still automatically change when one of the reward sources was removed. The biggest improvement in the algorithm will come from changing the form of the R̂π s estimator. For problems in which there is a single best solution, the KL-divergence measure seems to work well. However, we would like to be able to extend the load-unload result to the situation where the agent has a memory bit. In this case, the returns as a function of π are bimodal (due to the symmetry in the interpretation of the bit). In general, allowing each source's preference to be modelled in a more complex manner could help extend these results.*

While supportive, the entire piece can be referenced in the appendix for further reading and elucidation.

REWARDS & TRIGGERS

When are rewards activated? The system will be largely automated, using player performance data to manage the delivery of each type of reward. This does not mean we as the developer will not create a reward template for each Challenge. It merely means the system will be designed such that it will use both an adaptive algorithm and our hands-on design to engage and motivate the student. This list is not complete, but used as an example for discussion.

Actions that get you rewards	**Reward types**	**Category/Rationale**
completing things (tests, levels, journal entries) - required	points	participation
completing things w/ high or perfect score	bonus points	mastery
completing optional things	bonus points reward objects/badges	character: initiative
commenting on others' published items	badge	character: collaboration
publishing your work to others	badge	character: collaboration, mentoring
replaying games	badge	character: initiative

Type	Threshold	Action	Notes
Stars	% of task complete	70% = 1 Star 80% = 2 Stars 90% = 3 Stars 100% = 4 Stars	Used to incentivize completing each Topic to 100%. Useful for replayability.
Points	Objective Score	70% = 700 points 80% = 800 points 90% = 900 points 100% = 1000 points	We may consider a combo system in which successive 100% scores add a multiple to this number (2x, 3x, 4x etc)
Collectibles	Mission Objectives Content Use	Collectibles will be awarded for completing mission objectives and/or other in-game requirements.	Narrative based to further the story, reward content use (such as watching a video), or to create uber goals (earning a black panther)
Badges	Topic Completion	Topic Section = Badge Type Objective Score = Rank 70% = Rank 1 80% = Rank 2 90% = Rank 3 100% = Rank 4 Ex: Completing Photosynthesis and scoring a 90% would give the player the Photosynthesis Badge at Rank 3.	Students will earn badges as they make their way through each topic. The rank of the badge and the award will correspond to their performance on the Topic Assessments. Each rank will be represented by a new badge with visual and animated improvements. Players may retest to earn a higher rank badge.

COLLECTIBLES

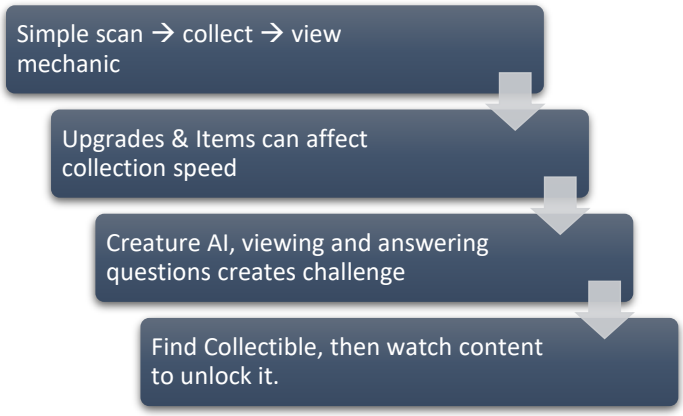
Capturing Collectibles

A reward for great play will be to let players recover Collectibles that can then be used to replicate that species in their pristine habitat. The Collectibles game will use the isometric view and maximize the player's learning by reinforcing information. It will use the existing mechanic of either rovers or avatars to move through an environment.

Collect Collectibles First

This becomes a strategic game in which the hunter and hunted move across a playing surface. This allows three distinct advantages:

- Uses an existing mechanic, 'scanning' things in the environment.
- Nonviolent play is more gender neutral, no violence associated with scanning.
- Easy to test a prototype and then scale up when fun.
- Has precedence in the game industry in Faunasphere. See case study (included at end of document)



The player must use their movement and any items, upgrades, or special powers to capture the target flora or fauna's Collectibles within a certain time limit. This is in addition to the normal mission Challenges.

In the case of flora, it will often be carried by insects, which are hard to trap. In the case of fauna, many species will be quick and hard to scan. It may require multiple scans to piece together one entire Collectibles segment for use.

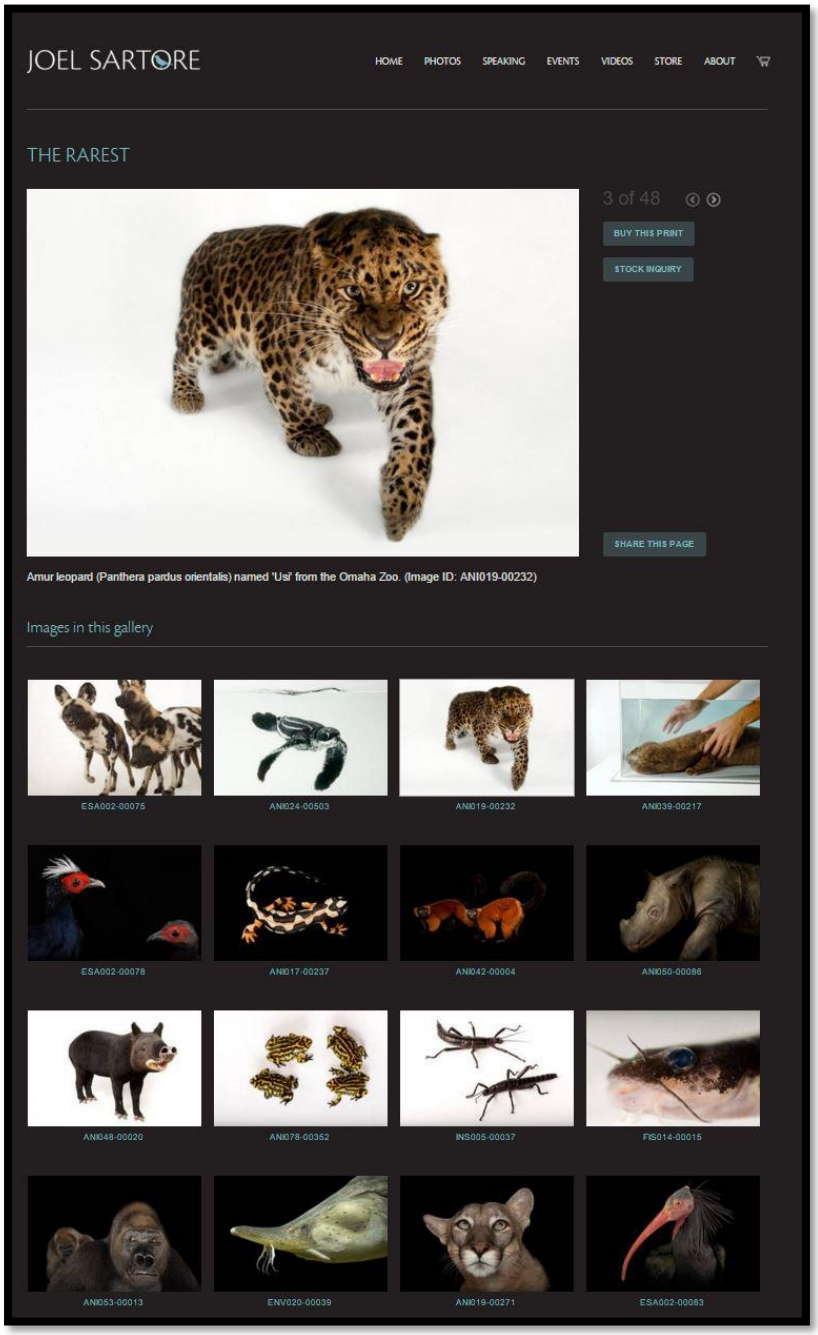
Most importantly, the collection of Collectibles gives the player a reason to explore the entire environment.

Unlock the Collectible Next

Once a player has captured a Collectible, it may be unlocked after the player has completed the requisite tasks associated with that Collectible. For example, unlocking the tiger Collectible might require the player to collect and watch all infocubes on Predation, and score a 90% or better on the post Assessment. Once those requirements have been met, the tiger Collectibles will unlock and the player may use it in their Habitat.

Some questions become immediately obvious and are answered below, however this is an open discussion and many more design decisions will have to be made after discussing the pros and cons of the reward system.

1. **Why collect Collectibles first?**
   By letting the player know what Collectibles they have, the motivation to unlock it is a powerful one we can use to promote mastery of a topic by understanding the rich media content associated with that topic. Students are less likely to 'click thru' a video if they know to unlock the tiger Collectibles they need to pay attention to the subject at hand.
2. **Why doesn't the Collectibles have to be associated with the Topic?**
   Because players all want something different. By giving them power over what they want to unlock for their Habitat, we are removing one hurdle to learning, and providing the player a sense of control over their learning direction and pace. By choosing their reward, they are being co-opted into learning.
3. **Can a teacher control / override this?**
   YES! Teachers can automatically reward students with points, stars, badges, or Collectibles for the completion of any task, giving them a powerful tool to drive learning and mastery.



Example Collectible - Joel Sartore Artwork

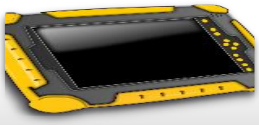
As an example, we could create collectible 'cards' (digital photos) of rare animals from Joel Sartore's work. Each collectible piece could be brought to life within the player's habitat, creating a menagerie of compelling creatures to motivate students to continue their learning.

Each digital photo here would represent an actual animal that would come to life within the player's habitat. The player will then learn how to keep that animal alive, using lessons learned from the various topics and by doing research using our in-game field guide.

Simple Habitat Play Example

Habitat Gameplay Narrative – This is an example that assumes the player has gathered many items and accessories through multiple successful Challenges. In other words, this is a mid-game example.

Player steps to build a Simple Habitat

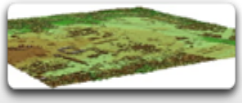


One icon is bouncing on the lower left. This is my DEC (name is tbd), with instructions and guidance for building a Habitat. This instructional tool is vital to keep the player informed of what to do next.

I choose an animal (Example: Tiger) from Collectibles I've earned from my Challenge. I earned this by collecting, watching, and participating in all content related tasks and have proven I understood the material (via assessment).

I select a Habitat Template from a number of pre-fabricated landmasses. Each is marked as more or less suitable for my tiger. I pick one that seems to be a close match (90% or better rating)

I click 'Auto-Complete' and Habitat grows. Advanced users can grow the Habitat manually.

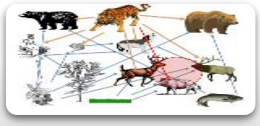


Ground layer becomes visible and interactable.

Surrounding the ground layer view is a new Toolbox Screen. Part of this screen will also include a 'Store' button which will take us to a new menu to purchase items for your Habitat using Points.

I insert the Collectibles into my Pristine Habitat matrix. The matrix is just a term for a blank landmass canvas. The Collectibles is 'seeded' into this blank canvas. Tiger appears where I drop the Collectible. From that area will appear a young tiger. This tiger will have needs, which will motiviate me to earn more Rewards and Collectibles to care for this tiger.

"Congratulations on getting your first apex predator, the Siberian tiger! Consult your DEC to ensure you're providing it with the best care…"



DEC teaches me what I need to create thriving flora & fauna, including Food Web and Trophic Levels. This sets up a vital part of the Motivation Loop, giving me something I want to collect to make my tiger happier/better. The parts I'm still missing will be greyed out but can be earned by doing more Challenges.

The DEC contains information necessary to successfully build a Habitat. This is just one example but we can scale this easily once we have determined budget and time constraints.

THE INTRINSIC REWARD – HABITAT

As the player invests more time and extrinsic rewards (points, stars, Collectibles) into their Habitat, the actual landscape will grow lush and fertile. The player will transform slowly from a 'collector' (I want something more for my habitat) into a 'customizer' (Player wants habitat to look a certain way unique to them).

Four Intrinsic Reward Drivers for Player Engagement

Our ultimate goal is for a player to be driven via intrinsic rather than extrinsic rewards. The reclamation and nurturing of a Habitat can be used to promote the four basic drivers of intrinsic player motivation:

1. **Meaningfulness**: Having a meaningful purpose
2. **Choice**: Having choice on how to fulfill that purpose
3. **Competence**: Performing competently
4. **Progress**: Making measurable progress

This desire to have meaning, choice, competence, and feel progress will be reflected in the design of the customizable the habitat. We will allow players to reflect personal taste by moving objects around, reshaping the land and water, and resculpting the terrain itself. We will make the cost of doing this minimal or zero, allowing the player to recast their island to their heart's content.

This becomes a powerful intrinsic reward, for the island transforms from a collection of earned objects to a canvas upon which the player has cast a reflection of themselves. This reflection space becomes a vital place for us to use other methods to reinforce learning, including asking questions about player choices that have no right or wrong answer.

By giving the player a place of self-expression (that can ultimately be shared with classmates) we create a powerful intrinsically motivating tool for student engagement.

BALANCING MULTIPLE SOURCES OF REWARD IN REINFORCEMENT LEARNING

- Programmer Section -

Christian R. Shelton Artificial Intelligence Lab Massachusetts Institute of Technology Cambridge, MA 02139 cshelton@ai.mit.edu

Abstract

For many problems which would be natural for reinforcement learning, the reward signal is not a single scalar value but has multiple scalar components. Examples of such problems include agents with multiple goals and agents with multiple users. Creating a single reward value by combining the multiple components can throw away vital information and can lead to incorrect solutions. We describe the multiple reward source problem and discuss the problems with applying traditional reinforcement learning. We then present an new algorithm for finding a solution and results on simulated environments.

1 Introduction

In the traditional reinforcement learning framework, the learning agent is given a single scalar value of reward at each time step. The goal is for the agent to optimize the sum of these rewards over time (the return). For many applications, there is more information available. Consider the case of a home entertainment system designed to sense which residents are currently in the room and automatically select a television program to suit their tastes. We might construct the reward signal to be the total number of people paying attention to the system. However, a reward signal of 2 ignores important information about which two users are watching. The users of the system change as people leave and enter the room. We could, in theory, learn the relationship among the users present, who is watching, and the reward. In general, it is better to use the domain knowledge we have instead of requiring the system to learn it. We know which users are contributing to the reward and that only present users can contribute. In other cases, the multiple sources aren't users, but goals. For elevator scheduling we might be trading off people serviced per minute against average waiting time. For financial portfolio managing, we might be weighing profit against risk. In these cases, we may wish to change the weighting over time. In order to keep from having to relearn the solution from scratch each time the weighting is changed, we need to keep track of which rewards to attribute to which goals. There is a separate difficulty if the rewards are not designed functions of the state but rather are given by other agents or people in the environment. Consider the case of the entertainment system above but where every resident has a dial by which they can give the system feedback or reward. The rewards are incomparable. One user may decide to reward the system with values twice as large as those of another which should not result in that user having twice the control over the entertainment. This isn't limited to scalings but also includes any other monotonic transforms of the returns. If the users of the system know they are training it, they will employ all kinds of reward strategies to try to steer the system to the desired behavior [2]. By keeping track of the sources of the rewards, we will derive an algorithm to overcome these difficulties.

1.1 Related Work

The work presented here is related to recent work on multiagent reinforcement learning [1, 4, 5, 7] in that multiple rewards signals are present and game theory provides a solution. This work is different in that it attacking a simpler problem where the computation is consolidated on a single

agent. Work in multiple goals (see [3, 8] as examples) is also related but assumes either that the returns of the goals are to be linearly combined for an overall value function or that only one goal is to be solved at a time.

1.2 Problem Setup

We will be working with partially observable environments with discrete actions and discrete observations. We make no assumptions about the world model and thus do not use belief states. x(t) and a(t) are the observation and action, respectively, at time t. We consider only reactive policies (although the observations could be expanded to include history). π(x, a) is the policy or probability the agent will take action a when observing x. At each time step, the agent receives a set of rewards (one for each source in the environment), rs(t) is the reward at time t from source s. We use the average reward formulation and so Rπ s = limn→∞ 1 nE [rs(1) + rs(2) + · · · + rs(n)|π] is the expected return from source s for following policy π. It is this return that we want to maximize for each source. We will also assume that the algorithm knows the set of sources present at each time step. Sources which are not present provide a constant reward, regardless of the state or action, which we will assume to be zero. All sums over sources will be assumed to be taken over only the present sources. The goal is to produce an algorithm that will produce a policy based on previous experience and the sources present. The agent's experience will take the form of prior interactions with the world. Each experience is a sequence of observations, action, and reward triplets for a particular run of a particular policy.

2 Balancing Multiple Rewards

2.1 Policy Votes

If rewards are not directly comparable, we need to find a property of the sources which is comparable and a metric to optimize. We begin by noting that we want to limit the amount of control any given source has over the behavior of the agent. To that end, we construct the policy as the average of a set of votes, one for each source present. The votes for a source must sum to 1 and must all be non-negative (thus giving each source an equal "say" in the agent's policy). We will first consider restricting the rewards from a given source to only affect the votes for that source. The form for the policy is therefore:

$$\pi(x,a) = \frac{\sum_s \alpha_s(x) v_s(x,a)}{\sum_s \alpha_s(x)}$$

Where π(x, a) = P s αs(x)vs(x, a) P s αs(x) (1) where for each present source s, P x αs(x) = 1, αs(x) ≥ 0 for all x, P a vs(x, a) = 1 for all x, and vs(x, a) ≥ 0 for all x and a. We have broken apart the vote from a source into two parts, α and v. αs(x) is how much effort source s is putting into affecting the policy for observation x. vs(x, a) is the vote by source s for the policy for observation x. Mathematically this is the same as constructing a single vote (v 0 s (x, a) = αs(x)vs(x, a)), but we find α and v to be more interpretable. We have constrained the total effort and vote any one source can apply. Unfortunately, these votes are not quite the correct parameters for our policy. They are not invariant to the other sources present. To illustrate this consider the example of a single state with two actions, two sources, and a learning agent with the voting method from

above. If s1 prefers only a1 and s2 likes an equal mix of a1 and a2, the agent will learn a vote of (1,0) for s1 and s2 can reward the agent to cause it to learn a vote of (0,1) for s2 resulting in a policy of (0.5,0.5). Whether this is the correct final policy depends on the problem definition. However, the real problem arises when we consider what happens if s1 is removed. The policy reverts to (0, 1) which is far from s2's (the only present source's) desired (0.5, 0.5) Clearly, the learned votes for s2 are meaningless when s1 is not present. Thus, while the voting scheme does limit the control each present source has over the agent, it does not provide a description of the source's preferences which would allow for the removal or addition (or reweighting) of sources.

2.2 Returns as Preferences

While rewards (or returns) are not comparable across sources, they are comparable within a source. In particular, we know that if Rπ1 s > Rπ2 s that source s prefers policy π1 to policy π2. We do not know how to weigh that preference against a different source's preference so an explicit tradeoff is still impossible, but we can limit (using the voting scheme of equation 1) how much one source's preference can override another source's preference. We allow a source's preference for a change to prevail in as much as its votes are sufficient to affect the change in the presences of the other sources' votes. We have a type of a general-sum game (letting the sources be the players of game theory jargon). The value to source s 0 of the set of all sources' votes is Rπ s 0 where π is the function of the votes defined in equation 1. Each source s 0 would like to set its particular votes, αs 0 (x) and v 0 s (x, a) to maximize its value (or return). Our algorithm will set each source's vote in this way thus insuring that no source could do better by "lying" about its true reward function. In game theory, a "solution" to such a game is called a Nash Equilibrium [6], a point at which each player (source) is playing (voting) its best response to the other players. At a Nash Equilibrium, no single player can change its play and achieve a gain. Because the votes are real-valued, we are looking for the equilibrium of a continuous game. We will derive a fictitious play algorithm to find an equilibrium for this game.

3 Multiple Reward Source Algorithm

3.1 Return Parameterization

In order to apply the ideas of the previous section, we must find a method for finding a Nash Equilibrium. To do that, we will pick a parametric form for Rˆπ s (the estimate of the return): linear in the KL-divergence between a target vote and π. Letting as, bs, βs(x), and ps(x, a) be the parameters of Rˆπ s ,

$$\hat{R}_s^{\pi} = a_s \sum_x \beta_s(x) \sum_a p_s(x,a) \log \frac{p_s(x,a)}{\pi(x,a)} + b_s$$

Rˆπ s = as X x βs(x) X a ps(x, a)log ps(x, a) π(x, a) + bs (2) where as ≥ 0, βs(x) ≥ 0, P x βs(x) = 1, ps(x, a) ≥ 0, and P a ps(x, a) = 1. Just as αs(x) was the amount of vote source s was putting towards the policy for observation x, βs(x) is the importance for source s of the policy for observation x. And, while vs(x, a) was the policy vote for observation x for source s, ps(x, a) is the preferred policy for observation x for source s. The constants as and bs allow for scaling and translation of the return. If we let p 0 s (x, a) = asβs(x)ps(x, a), then, given experiences of different policies and their empirical returns, we can

estimate p 0 s (x, a) using linear least-squares. Imposing the constraints just involves finding the normal least-squares fit with the constraint that all p 0 s (x, a) be non-negative. From p 0 s (x, a) we can calculate as = P x,a p 0 s (x, a), βs(x) = 1 as P a p 0 s (x, a) and ps(x, a) = p 0 s (x,a) a0 p 0 s (x,a0) . We now have a method for solving for Rˆπ s given experience. We now need to find a way to compute the agent's policy.

3.2 Best Response Algorithm

To produce an algorithm for finding a Nash Equilibrium, let us first start by deriving an algorithm for finding the best response for source s to a set of votes. We need to find the set of αs(x) and vs(x, a) that satisfy the constraints on the votes and maximize equation 2 which is the same as minimizing

$$\sum_x \beta_s(x) \sum_a p_s(x,a) \log \frac{\sum_{s'} \alpha_{s'}(x) v_{s'}(x)}{\sum_{s'} \alpha_{s'}(x)}$$

over αs(x) and vs(x, a) for given s because the other terms depend on neither αs(x) nor vs(x, a). To minimize equation 3, let's first fix the α-values and optimize vs(x, a). We will ignore the non-negative constraints on vs(x, a) and just impose the constraint that P a vs(x, a) = 1. The solution, whose derivation is simple and omitted due to space, is

$$v_s(x,a) = \frac{\sum_{s' \neq s} \alpha_{s'}(x)(p_s(x,a) - v_{s'}(x,a))}{\alpha_s(x)} .$$

vs(x, a) = P s 06=s αs 0 (x)(ps(x, a) − vs 0 (x, a)) αs(x) .

$$v_s(x,a) = \frac{\sum_{s' \neq s} \alpha_{s'}(x)(p_s(x,a) - v_{s'}(x,a))}{\alpha_s(x)} .$$

We impose the non-negative constraints by setting to zero any vs(x, a) which are negative and renormalizing. Unfortunately, we have not been able to find such a nice solution for αs(x). Instead, we use gradient descent to optimize equation 3 yielding Δαs(x) ∝ βs(x) P s 0 αs 0 (x) X a ps(x, a) P s 06=s αs 0 (x)(vs(x, a) − vs 0 (x, a)) P s 0 αs 0 (x)vs 0 (x, a) .

$$\Delta\alpha_s(x) \propto \frac{\beta_s(x)}{\sum_{s'} \alpha_{s'}(x)} \sum_a p_s(x,a) \frac{\sum_{s' \neq s} \alpha_{s'}(x)(v_s(x,a) - v_{s'}(x,a))}{\sum_{s'} \alpha_{s'}(x) v_{s'}(x,a)} .$$

We constrain the gradient to fit the constraints. We can find the best response for source s by iterating between the two steps above. First we initialize αs(x) = βs(x) for all x. We then solve for a new set of vs(x, a) with equation 4. Using those v-values, we take a step in the direction of the gradient of αs(x) with equation 5. We keep repeating until the solution converges (reducing the step size each iteration) which usually only takes a few tens of steps.

Cargo is loaded in state 1. Delivery to a boxed state results in reward from the source associated with that state. The left is the solution found. For state 5, from left to right are shown the p-values, the v-values, and the policy.

Transfer of the load-unload solution: plots of the same values as in figure 1 but with the left source absent. No additional learning was allowed (the left side plots are the same). The votes, however, change, and thus so does the final policy.

3.3 Nash Equilibrium Algorithm

To find a Nash Equilibrium, we start with $\alpha s(x) = \beta s(x)$ and $vs(x, a) = ps(x, a)$ and iterate to an equilibrium by repeatedly finding the best response for each source and simultaneously replacing the old solution with the new best responses. To prevent oscillation, whenever the change in $\alpha s(x)vs(x, a)$ grows from one step to the next, we replace the old solution with one halfway between the old and new solutions and continue the iteration.

4 Example Results

In all of these examples we used the same learning scheme. We ran the algorithm for a series of epochs. At each epoch, we calculated π using the Nash Equilibrium algorithm. With probability , we replace π with one chosen uniformly over the simplex of conditional distributions. This insures some exploration. We follow π for a fixed number of time steps and record the average reward for each source. We add these average rewards and the empirical estimate of the policy followed as data to the least-squares estimate of the returns. We then repeat for the next epoch.

4.1 Multiple Delivery Load-Unload Problem

We extend the classic load-unload problem to multiple receivers. The observation state is shown in figure 1. The hidden state is whether the agent is currently carrying cargo. Whenever the agent enters the top state (state 1), cargo is placed on the agent. Whenever the agent arrives in any of the boxed states while carrying cargo, the cargo is removed and the agent receives reward. For each boxed state, there is one reward source who only rewards for deliveries to that state (a reward of 1 for a delivery and 0 for all other time steps). In state 5, the agent has the choice of four actions each of which moves the agent to the corresponding state without error. Since the agent cannot observe neither whether it is:

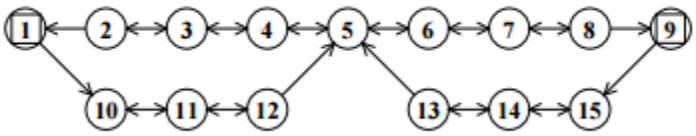
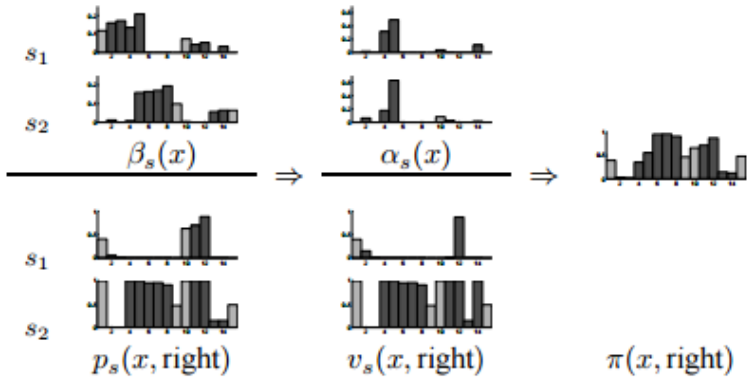


Figure 3: One-way door state diagram: At every state there are two actions (right and left) available to the agent. In states 1, 9, 10, and 15 where there are only single outgoing edges, both actions follow the same edge. With probability 0.1, an action will actually follow the other edge.

Figure 4: One-way door solution: from left to right: the sources' ideal policies, the votes, and the final agent's policy. Light bars are for states for which both actions lead to the same state. has cargo nor its history, the optimal policy for state 5 is stochastic.



The algorithm set all α- and β-values to 0 for states other than state 5. We started at 0.5 and reduced it to 0.1 by the end of the run. We ran for 300 epochs of 200 iterations by which point the algorithm consistently settled on the solution shown in figure 1. For each source, the algorithm found the best solution of randomly picking between the load state and the source's delivery state (as shown by the p-values). The votes are heavily weighted towards the delivery actions to overcome the other sources' preferences resulting in an approximately uniform policy. The important point is that, without additional learning, the policy can be changed if the left source leaves. The learned α- and p-values are kept the same, but the Nash Equilibrium is different resulting in the policy in figure 2.

4.2 One-way Door Problem

In this case we consider the environment shown in figure 3. From each state the agent can move to the left or right except in states 1, 9, 10, and 15 where there is only one possible action. We can think of states 1 and 9 as one-way doors. Once the agent enters states 1 or 9, it may not pass back through except by going around through state 5. Source 1 gives reward when the agent passes through state 1.

Source 2 gives reward when the agent passes through state 9. Actions fail (move in the opposite direction than intended) 0.1 of the time. We ran the learning scheme for 1000 epochs of 100 iterations starting at 0.5 and reducing it to 0.015 by the last epoch. The algorithm consistently converged to the solution shown in figure 4. Source 1 considers the left-side states (2–5 and 11–12) the most important while source 2 considers the right-side states (5–8 and 13–14) the most important. The ideal policies captured by the p-values show that source 1 wants the agent to move left and source 2 wants the agent to move right for the upper states (2–8) while the sources agree that for the lower states (11–14) the agent should move towards state 5.

The votes reflect this preference and agreement. Both sources spend most of their vote on state 5, the state they both feel is important and on which they disagree. The other states (states for which only one source has a strong opinion or on which they agree), they do not need to spend

much of their vote. The resulting policy is the natural one: in state 5, the agent randomly picks a direction after which, the agent moves around the chosen loop quickly to return to state 5.

Just as in the load-unload problem, if we remove one source, the agent automatically adapts to the ideal policy for the remaining source (with only one source, s0, present, $\pi(x, a) = ps0(x, a)$). Estimating the optimal policies and then taking the mixture of these two policies would produce a far worse result. For states 2–8, both sources would have differing opinions and the mixture model would produce a uniform policy in those states; the agent would spend most of its time near state 5. Constructing a reward signal that is the sum of the sources' rewards does not lead to a good solution either. The agent will find that circling either the left or right loop is optimal and will have no incentive to ever travel along the other loop.

5 Conclusions

It is difficult to conceive of a method for providing a single reward signal that would result in the solution shown in figure 4 and still automatically change when one of the reward sources was removed. The biggest improvement in the algorithm will come from changing the form of the $R\hat{}\pi$ s estimator. For problems in which there is a single best solution, the KL-divergence measure seems to work well. However, we would like to be able to extend the load-unload result to the situation where the agent has a memory bit. In this case, the returns as a function of π are bimodal (due to the symmetry in the interpretation of the bit). In general, allowing each source's preference to be modelled in a more complex manner could help extend these results.

Acknowledgments

We would like to thank Charles Isbell, Tommi Jaakkola, Leslie Kaelbling, Michael Kearns, Satinder Singh, and Peter Stone for their discussions and comments. This report describes research done within CBCL in the Department of Brain and Cognitive Sciences and in the AI Lab at MIT. This research is sponsored by a grants from ONR contracts Nos. N00014- 93-1-3085 & N00014-95-1-0600, and NSF contracts Nos. IIS-9800032 & DMS-9872936. Additional support was provided by: AT&T, Central Research Institute of Electric Power Industry, Eastman Kodak Company, Daimler-Chrysler, Digital Equipment Corporation, Honda R&D Co., Ltd., NEC Fund, Nippon Telegraph & Telephone, and Siemens Corporate Research, Inc.

References

1. *J. Hu and M. P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In Proc. of the 15th International Conf. on Machine Learning, pages 242–250, 1998.*
2. *C. L. Isbell, C. R. Shelton, M. Kearns, S. Singh, and P. Stone. A social reinforcement learning agent. 2000. submitted to Autonomous Agents 2001.*
3. *J. Karlsson. Learning to Solve Multiple Goals. PhD thesis, University of Rochester, 1997.*
4. *M. Kearns, Y. Mansouor, and S. Singh. Fast planning in stochastic games. In Proc. of the 16th Conference on Uncertainty in Artificial Intelligence, 2000.*
5. *M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In Proc. of the 11th International Conference on Machine Learning, pages 157–163, 1994.*
6. *G. Owen. Game Theory. Academic Press, UK, 1995.*
7. *S. Singh, M. Kearns, and Y. Mansour. Nash convergence of gradient dynamics in general-sum games. In Proc. of the 16th Conference on Uncertainty in Artificial Intelligence, 2000.*
8. *S. P. Singh. The efficient learning of multiple task sequences. In NIPS, volume 4, 1992.*

CASE STUDY: FAUNASPHERE (BIGFISH GAMES, 2009)

Achievements, Motivations and Rewards in Faunasphere

by Jason Begy, Mia Consalvo

Abstract

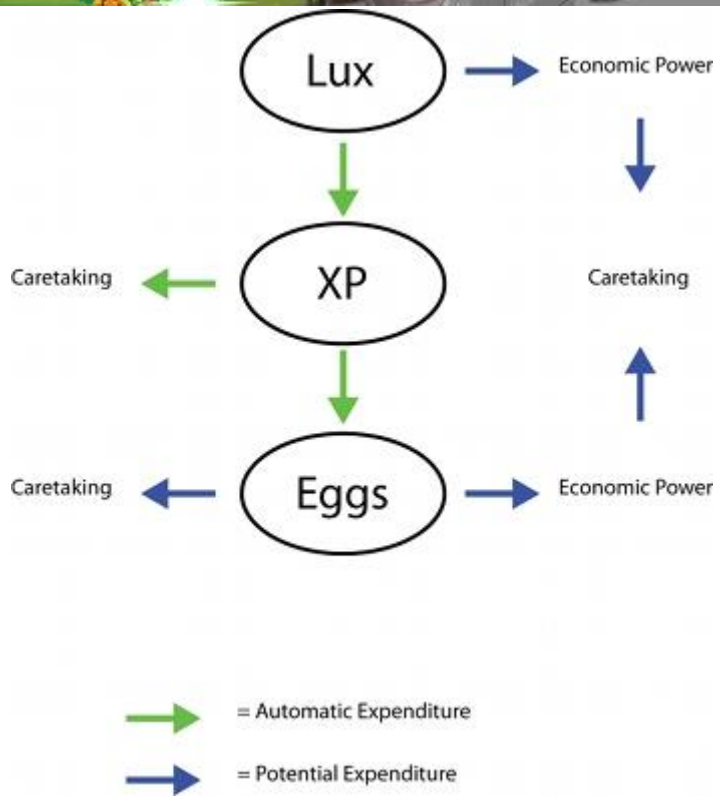
A persistent topic in MMO studies is player motivation: why do players play? Attempts to answer this question have resulted in several frameworks, but these are built from studies of games that are very similar to each other. MMOs tend to feature fictional worlds drawn from common fantasy and science fiction tropes, and gameplay tends to focus on continually improving one's personal power through leveling up a character and finding new equipment. This in turn has shaped how we interpret player activity. However, casual games developer Big Fish Game's first MMO, Faunasphere, breaks several of these tropes.

Players are cast as "caretakers" of small animals, referred to as "fauna," and must guide them through the world while improving their home and keeping them happy.

The game is almost entirely nonviolent, and the reward for leveling up is not a gain in personal power, but rather an egg used to hatch a new creature to care for.

This study examines the players of Faunasphere to determine who they are, what their backgrounds are, and what motivates them to continue playing.

We argue that while their motivations seem to fit traditional frameworks, the nature of player activity (as defined by the game's fiction) is significantly different. This is due in part to the game's reward structure, which is designed to continually reinforce the player's position as caretaker: every reward in the game either can be or must be used to further care for one's fauna. We conclude with a call for greater contextualization when discussing concepts such as reward and achievement: as



Faunasphere shows, the fictional world of a game has a strong impact on how players interpret their in-game activities.

Players of Faunasphere are called Caretakers, and their job is to raise diverse types of “fauna.” Players begin the game with one fauna, which the player must use to zap pollution (rather than kill monsters), which appears as cube-like entities in the game. Each zapped cube yields money, experience points and items. Once a certain level of XP is reached the fauna gains a level, and also lays an egg. Players can level a fauna to a maximum of 20 through the zapping of pollution as well as the completion of various goals (quests in MMOG-speak) in the gameworld.

The primary reward in Faunasphere is Lux, which is the basic currency. Lux is earned by zapping pollution or completing goals, and is used to purchase goods from shops or other players’ totems. Whenever Lux is earned, an equivalent amount of experience points (XP) are earned at the same time (experience points are not lost when Lux is spent).

When a fauna earns a predetermined amount of experience points it gains a level--much as in other MMOs--and lays an egg. Eggs are used to hatch new fauna, and can be bought, sold, gifted and traded just like other items in the game.

At the highest level, Faunasphere’s reward structure can be thought of as a series of deterministic relationships. Earning Lux always entails earning XP, which always entails earning new eggs. This process is illustrated here.

One may read the entire case study here: http://gamestudies.org/1101/articles/begy_consalvo