Issues of Autonomous Character Design
(The Truth About Catz and Dogz)

From the proceedings of the 1997 Computer Games Developer Conference, Santa Clara, CA

By Ben Resner, Adam Frank, Andrew Stern.
benres @aol.com

What is an Autonomous Character?

Traditional computer games have centered around characters for years. But these are not autonomous characters, they're merely puppets controlled by the user. The dexterity of the game's character is directly proportional to the dexterity of the user. For example, the faster someone can hit the fire button, the more rounds of ammo the character can fire off.

An autonomous character is not directly controlled by the user. This should not imply they are independent of the user, for their behavior is strongly influenced by what the user does. The user and the character form a relationship, where each one learns from the other.

In traditional games, the user is represented in the game world as the character, either by an image of the character, or the character's point of view. With an autonomous characters, the user is represented as the user himself, usually via the arrow cursor or some other independent interface element.

This distinction is central to the design of the Petz products, starting with Dogz, then Catz, and most recently Oddballz. For example, in Catz and Oddballz, the pet has the same access to the supply shelf as the user. This reinforces the relationship between the user and the pet. The toys and other objects in the Petz playpen are for both the user and the pet. The user does not have special access to items in the Pet's world.

Disney, not Minsky

For a computer pet or other autonomous character to be believable it must show personality. To that end, we've focused on Artificial Personality (AP), rather than Artificial Intelligence (AI). AI is clearly superior at solving actual problems, such as learning mazes, recognizing faces, or sorting items according to color, size, or some other constraint. But AI rarely looks to the user like intelligence. Users make judgments of intelligence based on how the pet acts, not on the final outcome of their actions. For example, a dog that struggles to figure out how to open a cabinet but ultimately fails is much more appealing to a user than a dog that figures out the system immediately. While the latter dog has a superior AI engine, and is more "intelligent", the former will have much more warmth.

To this end, Petz design have been much more influenced by Disney than by Minsky. Timing is the central element of comedy, and it's no different for autonomous lifelike characters. AI focuses on the result, neglecting timing and performance. AP tools center around the ability to tightly control timing. Many variations and alternate scripts for the same goal are included in the Pet to avoid any one "solution" becoming stale. It's more important for the pet to look intelligent than to actually be intelligent.

Storytelling

One of the most powerful aspects of the user-pet relationship is the user's ability to tell stories about the Pet. We receive lots of mail from customers where they narrate experiences with their pet far beyond what any AP or AI engine could ever hope for. These are stories full of complex and sophisticated relationships developed over long periods of time that are simply not part of the shipping code.

Users tell these stories because the pet looks like a pet and acts like a pet. Petz development has been shameless about taking advantage of this. Just as children will tell stories about their stuffed animals as if they were real, adults will tell stories about their computer pet, as if it was real. The only difference between adults and children is adults need to be tricked into telling their story.

A fun "party trick" is to ask someone to make up a story. Children have no problem with this kind of make-believe, and start talking immediately. Most adults, however, will stammer, saying: "Oh, I don't think so, I can't tell stories". The reluctant adult is then told: "OK, I'm thinking of a story. Why don't you ask me yes or no questions about the story I'm thinking of". Provided the person is still game, questions usually flow freely. "Does it have a happy ending?" "Is it about a baby?", or "is it about someone at the Computer Games Developer Conference?".

Questions are answered according to a very simple algorithm. If the question ends in a consonant, answer "yes". If it ends in a vowel, answer "no", and if it ends in a "y", say "maybe". If there are too many "no's" in a row, answer "yes". This avoids negativity, which is discouraging.

The adult has essentially been tricked into telling a story. Never mind that it's being told as a series of questions, the adult generating a narration totally on his own. But before this could happen, the adult had to be convinced it wasn't their story, thus removing all risk of embarrassment.

With autonomous Petz, this "risk of storytelling" is similarly removed. Adults feel silly telling stories about stuffed animals because they've been taught they should know better. But the vast majority of adults is unaware of the limits of artificial life on an average desktop computer, and will accept a behavior engine far more powerful than what an average desktop computer can support. They don't feel childish narrating complex stories about their pets because they're convinced the pet is the origin of the story, and not their own mind. We focus on creating the illusion of desktop life.

Creating a pet which allows this flexibility is a lot of work. A pet with too much random behavior will appear disconnected. Alternatively, too little randomness creates a robotic and repetitive pet. Tuning the personality between these two extremes is a large part of the art of realistic personality development. We use what we call "constrained randomness" to find this balance.

One of the design goals of Petz was to never penalize the user for doing something wrong. During product development, when we would watch users using the Pet, we're often tempted to say: "Don't do that" or "You're doing that wrong". This wasn't a shortcoming of the user, it's a shortcoming of the product. The issue isn't how to educate the user, but how to design the pet so that the user's offending action will have some effect on the pet. Every interaction should increase the user's ability to tell a story about the pet.

For example, petting is an excellent means for user to bond with the pet. Our first few attempts at petting were awkward, and it was difficult to for users to get the pet to respond. Instead of attempting to educate the user about how to pet a computer pet, we had to continually tweak the petting algorithm to allow for all kinds of users to have successful petting interactions. Often this meant reducing the sensitivity of petting in order for it to work with a wider audience.

We're in Show Biz

Computer Petz is about putting on a show with high entertainment value. Just as writers often remark their characters have taken on a life of their own, the same must happen with computer Petz. Computer Petz must have distinct personalities, recognizable from each other.

Before actual coding for a pet is started, a personality worksheet is generated. This drives all subsequent design decisions ? ? how does the pet walk, how does the pet respond to danger, and so on. Instead of "making it up as we go along", we have a fixed reference to guide all decisions points. This makes sure the pet has a consistent personality towards all objects in its world.

Direct Interaction -- Rendered vs. Cell Drawn

A key component of autonomous characters is direct interaction. Users need to feel as if they're actually touching a real pet.

To this end, autonomous characters must be able to transition from any state to any other state in a very short amount of time. When the user double-clicks to gain the Pet's attention, the response must be immediate. If the engine takes a few frames to respond, the effect is lost.

For this reason, Petz uses a real-time rendering engine rather than cell-drawn animation. While cell-drawn animation usually looks much better on a frame-by-frame basis, and is much easier to control, it has very limited variation, and rapid transitions are difficult.
By using real 3D animations, the pet can be viewed from any angle. We also use layering to combine two animations into one. For example, this allows the pet to walk while turning it's head to keep its gaze fixed to a moving object. Additionally, we use motion scaling and tweening further modify animation data. We also have control over eyelid height and eyelid tilt, extremely powerful tools for communicating the pet's emotional state to the user.

All these programmatic animation effects create a very lifelike pet. The user can play with the pet for an hour, and never see the exact same animation frame twice. Users are very keen to repetition, and once they see the same animation too many times, the pet ceases to be alive, and simply become a robot.

Direct interaction allows the user to actually feel like they've reached into the computer, and are touching a living animal. For many people, the computer pet idea doesn't "click" into place until they have a chance to grab or stroke the pet themselves. They'll watch a demo with a smile, but never actually laugh until they get to grab or stroke the pet themselves.

"Home-Grown" Versus "Off The Shelf"

Critical to the bonding between user and autonomous character is uniqueness. Much effort has gone into making each pet slightly different from each other. Users can color their pet, and teach them tricks. By lavishing or withholding attention or food, the user can change the pet's personality. Every user must be made to feel their pet is unique and special. If my friend and I adopt a pet of the same breed on the same day, after two weeks of playing, the pets will be noticeably different.

A key to this bonding is the adoption process. The user carefully selects their pet, which starts life as a puppy or kitten, and grows into an adult over the next few weeks. We use the real-time render effects described above to show the pet in any stage of maturity, from newborn to adult. Most people are naturally drawn to animals with big eyes and big paws, and we've found it's no different with computer Petz.

The user doesn't get a fully realized cartoon character with a predetermined name, and hardwired quirks. The user only gets a personality template. The user is responsible for naming the pet, and developing the idiosyncrasies that is one of the hallmarks of pet ownership.

With an interactive Bugs Bunny, for example, users would not expect to alter Bug's personality much, or see him grow and change over time. Bugs comes with established personality traits, and most users would not try and go outside these bounds. This may increase play-value in the short term because there's very little learning curve. But it's difficult to form a relationship with an inelastic personality.

With generic pets and pet breeds, the limits are much looser. People come into Petz with strong expectations of how a cat or dog acts, and even how different breeds act, but not much more. People expect to be able to influence the personality of their puppies and kittens.

One of the reasons we chose to do Dogz and Catz first is because of the expectations users bring to the product. When the user sees a ball and a dog, it's pretty clear what to do.

Death & Responsibility

Perhaps the largest design issue with autonomous characters is death. Should autonomous characters die? To many, a character can never be alive unless the user knows it will eventually die. Immortality cheapens the experience by not making the time precious.

After many debates we decided against death. While we're trying to create a realistic software pet, we're still software. Early focus tests and product feedback has made it very clear most users will be scared away by responsibility towards the pet, or the knowledge the pet will eventually die and cause them grief. While a bond does form between the user and the pet, the majority of the experience is still about fun and entertainment, and not responsibility.

Most people are aware that ill-behaved hyper puppies that gnaw furniture and mess up the floor, with proper training, will eventually turn into loyal and well-behaved companions. Imagine installing a piece of software that did similar things to your computer -- slowed performance, mangled or deleted files, and caused crashes. How long would it be before this software was deleted from the computer? It's way premature to expect people to tolerate the downside of a hyper-realistic computer pet in order to get to the reward.

As the power of computer pets grows, this will change. People will come to expect more, and thus be willing to work harder to get there. And this hard work on the part of the user will become a powerful tool for bonding with the pet.