Андрей Родионов — Своя игра

> ИГРА — вид непродуктивной деятельности, мотив которой заключается не в её результатах, а в самом процессе. В истории человеческого общества переплеталась с магией, культовым поведением и др. Свойственна также высшим животным.
>
> *—Советский энциклопедический словарь*

С тех пор, как я начал заниматься программированием, мне не раз приходилось сталкиваться с различными компьютерными играми — такими, которые могли общаться с играющим только при помощи неоновых лампочек на пультах старых ламповых ЭВМ, и играми на суперсовременных компьютерах, больше похожими на интерактивный художественный фильм с элементами мультипликации, сложными видеоэффектами и многоканальным звуковым сопровождением. Но ещё интереснее оказалось самому создавать такие игры.

Многое захватывает в такой работе. И не столько сам процесс тестирования создаваемой игры, сколько разработка игровой вселенной, её проектирование и реализация. Когда можно слить воедино сценарий, графику, музыку, искусно задуманный и умело запрограммированный алгоритм — создать единый фантастический мир, живущий по законам, которые ты же для него и придумал.

Можно почувствовать себя почти всемогущим. Можно даже вообразить, что ты стал немного ближе к нашему Творцу и стал чуть лучше понимать его. «Limitation is only Your Imagination!» («Ограничение — только ваше воображение!») Эта фраза, использованная в своё время для рекламы музыкальных синтезаторов, на мой взгляд, как нельзя лучше подходит и для разработки компьютерных игр.

За долгие годы накопилось довольно много мыслей о программировании в целом и о разработке компьютерных игр в частности. Одни из них уже обсуждены в беседах с друзьями и коллегами, другие легли в основу этой статьи. Спасибо всем, кто терпеливо меня выслушивал и иногда давал советы.

Классификация компьютерных игр

Авторская классификация

Современные компьютерные игры я разделил бы прежде всего на два больших класса:

- для игровых автоматов;
- для индивидуального (домашнего) пользования.

Начнём с игр для игровых автоматов. Игровым автоматом будем называть устройство, предназначенное для сбора денег за предоставляемую возможность играть с этим устройством. Такие автоматы обычно устанавливают в общедоступных местах или специальных игровых залах. Заметим, что к этому классу не относятся специализированные игровые компьютеры, приобретаемые для индивидуального домашнего использования.

Игры, предназначенные для игровых автоматов, отличаются в первую очередь тем, что для накопления возможно большего количества монет (жетонов) число играющих с автоматом людей (или попыток одного и того же человека играть многократно) должно быть также как можно больше. Таким образом, игры для этих автоматов должны осуществлять жёсткое «давление» на игрока по времени и активную стратегию на «выбрасывание» играющего из игры. При этом сложность каждого нового этапа игры должна возрастать настолько, чтобы среднее время даже для опытных играющих не становилось слишком большим.

Кроме того, игры этого класса должны удовлетворять и некоторым другим требованиям: *

- первая стадия игры должна быть относительно простой и одновременно привлекательной для новичков, но при этом и весьма непродолжительной — новичок должен «выбрасываться» достаточно быстро, а опытный игрок — не задерживаться надолго;

- логическая схема игры должна быть реализована «жёстко» — практически без элементов случайности, так, чтобы играющий во время повторной игры мог использовать свой опыт прохождения всех предыдущих стадий игры для достижения новой стадии;
- игра должна «поощрять» игрока при завершении очередного этапа — оригинальной графикой, музыкой, а при достижении важных промежуточных стадий или финала — «призовой игрой». Это важно также для привлечения и поддержки «болельщиков», которые обычно находятся рядом с играющим и являются потенциальными новыми игроками;
- игра должна быть популярной длительное время, так как стоимость игрового автомата выше стоимости персонального компьютера, а частая смена работающих в нем программ обычно не предусматривается.

Некоторые автоматы выпускаются с управлением, рассчитанным только для какой–либо конкретной игры, например, кабина автомобиля или самолёта с имитатором наклона кабины, штурвалом, педалями и тремя экранами, расположенными панорамно.

Чаще всего, перед тем как оказаться в игровом автомате, базовая версия программы проходит «обкатку» в качестве игры для домашнего пользования, и только имея устойчивый высокий рейтинг в компьютерных журналах и пройдя специальную экспертизу фирм–изготовителей игровых автоматов, игра попадает туда, куда (возможно) и предназначалась первоначально. Окончательный вариант игры (то есть уже для автомата) обычно обладает более изощрённой графикой, звуком и другими эффектами, которые отсутствуют в базовом варианте игры для персональных компьютеров.

Однако фирмы–изготовители игровых автоматов разрабатывают и свои собственные специальные игры (часто вместе с уникальным оборудованием для них), которые не встречаются на персональных компьютерах и существуют только в версиях для игровых автоматов. К классу игр для игровых автоматов относятся, в первую очередь, так называемые «аркадианские» игры (Arcadian games), в которых приходится много стрелять из бластеров, пушек, сражаться различными способами со всевозможными врагами или спасаться от разнообразных монстров, пытающихся вас съесть. Действие может разворачиваться в любом времени или месте или вообще неизвестно где и когда. Цель всегда благородна: спасти кого–то (но, в первую очередь, самого себя), победить врага, пытающегося завоевать, разрушить, съесть и пр.

Игры этого класса могут быть выполнены с использованием великолепной графики, обладать прекрасным музыкальным и речевым сопровождением и выглядеть почти как художественный фильм, но при этом все равно являться играми, где надо все время «спасаться» и «побеждать», причём в том темпе, который диктует компьютер.

Во вторую очередь, я отношу к классу «игр для игровых автоматов» абстрактные игры типа Pacman и Tetris, которые, тем не менее, подчиняясь «закону игрового автомата», постепенно уменьшают время на обдумывание ситуации и сводят время реакции играющего к времени опроса компьютером органов управления игрой. Темп игры здесь также диктуется компьютером. Когда время реакции опытного играющего становится соизмеримым с временем опроса органов управления игровой программой, компьютер начинает реализовывать свою собственную выигрышную стратегию в ущерб приоритетности выполнения команд игрока.

Наконец, среди игр для игровых автоматов попадаются высококачественные трёхмерные тренажёры (в частности, уже упоминавшиеся автомобили и самолёты), требующие, в отличие от предыдущих игр, высокопроизводительных центрального и видеопроцессоров, специального оборудования и органов управления. Обычно они в меньшей степени подчинены стратегии скорейшего «выбрасывания» игрока, так как требуют более продолжительного освоения. Но это скорее исключение, чем правило. Постоянный спрос на такие игры поддерживается в основном из–за того, что они трудно воспроизводимы на обычных персональных компьютерах с недорогими графическими картами и стандартными периферийными устройствами управления.

Таким образом, игры для игровых автоматов (независимо от того, реализована игра действительно на игровом автомате или на универсальном компьютере) характеризуются тем, чти доминантой в них является возрастающие темп и сложность игры, которые постоянно диктуются компьютером. При этом вся воля и все умение игрока подчиняются одной–единственной цели — «как можно дольше продержаться».

Время для обдумывания и выбора стратегии и тактики игры практически отсутствует, но оно на самом деле и не требуется, так как каждая ситуация жёстко «зашита» в самой игре и в точности повторяет сама себя при повторении игры с самого начала. Помочь играющему может лишь одно — опыт преодоления предыдущих стадий при многократных попытках доиграть в такую игру до конца.

Здесь я позволю себе небольшое личное замечание: игры этого класса никогда меня не только не привлекали, но, скорее, наоборот — отталкивали, так как любое давление будило во мне протест. А в этих играх на меня давил мой собственный компьютер или игровой автомат!

Для себя я условно называю эти игры «злыми» по отношению к играющему.

Перейдём теперь ко второй группе: «игры для индивидуального (домашнего) использования». Игры этого класса, которые я условно называю «добрыми» по отношению к играющему, гораздо более разнообразны, чем «злые». Однако во избежание недоразумений сразу исключим из этого класса все игры, относящиеся к разновидности «злых», хотя вы, может быть, играли в них на своём персональном компьютере, а вовсе не на игровом автомате. Может быть, эти игры ещё проходят «обкатку» у пользователей или они уже есть в игровых автоматах, но вы об этом не знаете?

Напомню ещё раз, что «злые» игры легко отличить от «добрых» по степени «давления», оказываемого ими на играющего. В «злых» играх «давление» — самоцель и доминанта всей игры, так как ее истинной задачей является как можно более частая смена игроков для сбора денег; а в «добрых» — это лишь одно из множества средств для взаимодействия с играющим.

Конечно, и в «доброй» «домашней» игре могут быть ограничения по времени либо по каким—либо ресурсам или в виде активно нападающих врагов, но это «давление» почти всегда можно смягчить, а иногда даже и полностью его избежать, так что игра может продолжаться длительное время, давая возможность размышлять, спокойно принимать продуманные решения, а иногда и сохранять на внешних носителях промежуточные стадии игры для того, чтобы позже продолжить её с желаемого момента.

Общепринятая классификация

Приводимое ниже деление игр на типовые (жанровые) классы, хотя и используется достаточно широко во всем мире, является весьма условным, так как многие игры обладают сразу несколькими классификационными признаками, а иногда включают элементы обучения, программирования, разнообразные средства мультимедиа и многое другое. При рассмотрении типовых (жанровых) классов попытаемся спроектировать их на классы «добрых» и «злых» игр.

В условном классе «добрых» игр находятся практически все так называемые «приключенческие» игры (Adventure), где цель известна, но не известны конкретные способы ее достижения. Обычно играющему предоставляется возможность исследовать окружающий его мир, собирать и использовать различные предметы, общаться с окружающими его существами и т.д. Естественно, что такая исследовательская деятельность не терпит суеты и спешки, требует времени, внимания, размышлений и, следовательно, никак не может сочетаться с функциональным назначением игрового автомата (собрать как можно больше денег).

Разновидностью «приключенческих» являются «ролевые» (от слова роль) игры (RPG — Role Playing Games). В них играющий может выступать в качестве любого персонажа игры, а за остальных в это время играет компьютер. Особо популярны в качестве сюжетов для таких игр всевозможные миры класса «Фэнтези» и различные детективные расследования. Особую популярность завоёвывают в последнее время игры класса mRPG (Multipleer Role Playing Games) в которых присутствуют сразу много реальных игроков, взаимодействующих друг с другом на общем сетевом игровом сервере. На основе таких игр некоторые писатели даже сочиняют свои собственные романы.

Интересным подклассом в классе «добрых» игр являются различные «стратегические» игры (Strategy Games), из самого названия которых уже следует, что играть в них надо не торопясь и хорошо продумывая свою собственную стратегию. Примером могут служить стратегические планы ведения войн, разработки экономической политики и др.

«Симуляции» (Simulations) включают в себя моделирование как относительно небольших объектов (катеров, подводных лодок, самолетов, космических кораблей), так и более крупных (отдельных предприятий, городов, стран, планет, а также целых планетных и звёздных систем). Во всех случаях играющему предоставляется возможность добиваться каких–либо целей путём определённых воздействий на различные объекты и подсистемы, обладающие сложным поведением, и наблюдать за их реакцией в ответ на такие воздействия. Эти игры также обычно требуют к себе внимательного и вдумчивого отношения и чаще всего попадают в класс «добрых».

«Спортивные» игры (Sport Games) часто требуют хорошей реакции и быстроты действий играющего, поэтому оказываются в зависимости от реализации и среди «добрых» и среди «злых» игр. Примерами могут служить всевозможные «олимпиады», «многоборья», не слишком сложные «авто– и моторалли», «горные лыжи», «теннис», «гольф» и др.

Компьютерные реализации традиционных настольных игр (Board Games) чаще всего принадлежат к классу «добрых» (шахматы, го и т.п.). Карточные игры в зависимости от степени сложности и конкретной реализации могут попадать и в «добрый», и в «злой» классы.

Головоломки (Puzzles) чаще всего принадлежат к классу «добрых» игр, так как обычно предполагают определённые размышления.

Абстрактные игры типа Tetris с одинаковым успехом могут попадать в оба класса в зависимости от выбранных критериев окончания игры.

Уже упоминавшиеся «аркадианские» игры практически полностью относятся к классу «злых».

Обучающие игры (Learning Games) могут оказаться в любом из приведённых выше типовых классов, но при этом они преследуют конкретную цель — научить играющего чему–либо полезному: быстрому счету, чтению электронных схем или дорожных знаков, иностранному языку, печатанию на машинке, оптимальному взаимодействию с деловыми партнёрами или учреждениями и пр. Конечно же, почти все обучающие игры принадлежат к классу «добрых», так как предполагают вдумчивое и целенаправленное использование их для обучения. Отдельный подкласс обучающих игр образуют «тренажёры», выполняемые чаще всего в типовом классе «симуляции».

Приведённая выше классификация не претендует на полноту, но позволяет легче ориентироваться в новых, появляющихся на рынке играх. Тем не менее, следует ещё раз обратить внимание читателя на то, что в современных сложных компьютерных играх часто соседствуют и могут пересекаться элементы игр разных классов, что затрудняет их точную классификацию.

Особенности рынка игровых программ

Рынка игр для игровых автоматов практически не существует, поскольку игры для них, как уже было замечено выше, часто создаются, появляются, продаются и испытываются в первую очередь на рынке программ для ПК. Кроме того, фирма разработчик обычно не заинтересована в том, чтобы аналогичные игровые автоматы создавались и продавались другими изготовителями, кроме неё самой.

Чем же этот рынок интересен для разработчика компьютерных игр? Прежде всего тем, что он практически ненасыщаем. Большинство игр рано или поздно надоедают играющим, и в них перестают играть. На смену им приходят новые игры, которые, в свою очередь, вытесняются следующим поколением игр, и так далее...

Неотвратимость появления новых игр гарантируется, с одной стороны, необходимостью создания все новых и новых развлечений для заполнения досуга людей, а с другой стороны — появлением все более совершенных и мощных персональных компьютеров, позволяющих даже старые игровые идеи поворачивать новой и подчас неожиданной стороной, обогащать дополнительными эффектами, а иногда и наполнять совсем другим содержанием.

Естественно, что разработчиков компьютерных игр устраивает такое положение дел, поскольку оно всегда гарантирует спрос на их работу при условии конкурентоспособности их продукции и соответствующей охраны авторских прав.

Помимо этого, опытные разработчики оказываются заинтересованными также и в том, чтобы их продукция после завершения основного этапа продаж каждой отдельно взятой игры не блокировала бы продажу следующих за ними новых игр, то есть устаревала бы достаточно быстро. Для этого значительное количество игр сознательно проектируется так, что, будучи единожды пройденными от начала до конца, они в значительной степени теряют свою привлекательность из—за ставшей уже известной оптимальной последовательности действий. Этого добиваются обычно увеличением количества «жёстко» запрограммированных ситуаций и исключением или сведением к минимуму элементов вероятностного порождения мира игры.

Покупатель же, со своей стороны, хочет, чтобы купленная им игра (как и любая другая им приобретённая вещь) не теряла для него своей привлекательности как можно дольше!

Здесь, как мы видим, интересы разработчика игр и играющего существенно противоречат друг другу.

Это противоречие может смягчаться или отсутствовать в следующих случаях:

- если разработчик данной игры не относится к разработке компьютерных игр как к источнику постоянного заработка или, вообще, не рассматривает данную игру в качестве коммерческого продукта (разработчик–—непрофессионал)
- если разработчик–профессионал делает игровую программу в первую очередь «для самого себя».

Первая ситуация (разработчик–непрофессионал) встречается довольно часто. Каждый из нас может ознакомиться с образцами таких любительских игр, просматривая различные public domain (программы для бесплатного публичного копирования и использования). Обычно такие игры обладают маловыразительными графикой и звуковым сопровождением, недостаточно удобны в управлении, а иногда обладают и непонятным сценарием.

Вторая категория игр (разработчик–профессионал делает игру для себя) встречается сегодня крайне редко. Мне известно всего лишь несколько таких игр. Эти игры отличаются прежде всего тем, что, обладая великолепными художественными и алгоритмическими решениями, они при каждом новом запуске предлагают играющему

достаточно богатую, но несколько отличную от предыдущей игровую вселенную. При этом детали этой вселенной каждый раз немного изменяются случайным образом. Это очень точно взвешенное соотношение детерминизма и случайности иногда может создавать такое сочетание разнообразия и неожиданности, которое способно сделать игру практически «бессмертной». Однако соблюсти этот тонкий баланс, придающий игре «бессмертные» свойства в глазах не только автора, но других играющих, зачастую бывает нелегко и профессионалу, так как большинство игр со сложной и интересной игровой вселенной весьма склонны к вырождению либо в слишком запутанные и чересчур сложные для освоения другими игроками, либо в «хаотические».

Вообще говоря, разработка игровых вселенных связана, на мой взгляд, с очень серьёзными философскими и религиозными проблемами, но об этом я скажу несколько позже. А пока зададим себе вопрос:

С кем или с чем мы играем?

Ответ может быть тривиальным: «С компьютером!».

Или чуть сложнее: «С программой, реализующей алгоритм игры».

Но я предпочитаю развёрнутый и, как мне кажется, более содержательный ответ: «Каждый раз мы играем с некоторой вселенной, сконструированной авторами игры, которая состоит из объектов, участвующих в данной игре. Эти объекты подчиняются определённым законам, описывающим их поведение, и могут взаимодействовать как друг с другом, так и с играющим. Законы и объекты этой вселенной описываются алгоритмами и данными, совокупность, которых реализована в виде игровой программы».

Несмотря на кажущуюся сложность такого ответа, он применим к любым компьютерным играм, и каждый может убедиться в этом сам, вспомнив известные ему игровые программы.

Но его особая ценность, на мой взгляд, заключается в том, что с его помощью можно попытаться разобраться в том, чем же является компьютерная программа вообще и игровая программа а частности.

Что же такое компьютерная программа?

Для начала попробуем понять, чем принципиально отличается практически любая компьютерная программа, предусматривающая ввод (управление), обработку и вывод (отображение) данных, от других продуктов человеческого творчества.

В первую очередь, необходимо отметить, что творческие процессы в любой области человеческой деятельности имеют много общего. Любая творческая деятельность в области науки, литературы, изобразительного искусства, кино или музыки состоит из некоторых «озарений», всегда появляющихся непредсказуемо и внезапно, и из систематического «конструирования» и синтеза целого из этих отдельных «озарений», которое называется мастерством или школой.

В разных произведениях даже одного и того же автора соотношение элементов «озарений» и «конструирования» может быть различным. И все же не оно является критерием ценности конечного продукта творчества — ценность определяется цельностью и внутренней непротиворечивостью нового произведения, будь то литературная поэма или новая физическая теория. В любом случае происходит «отчуждение» результата работы духа и интеллекта автора в некоторый конечный продукт, доступный для понимания и употребления другими членами человеческого сообщества.

Не является в этом смысле исключением и одна из самых молодых областей приложения творческих способностей человека: область разработки программ для компьютеров. Работа в этой области точно так же состоит из «озарений» и «конструирования», качество конечного продукта зависит от мастерства автора и представляет тем большую ценность для окружающих, чем более цельным и внутренне непротиворечивым будет результат работы. Точно так же в конечном продукте присутствует «отчуждаемая» часть работы духа и интеллекта автора, но имеется и одно, очень существенное отличие: «отчуждённый» продукт, применительно к компьютерным программам, обладает поведением, которое автор описывает в виде алгоритма работы программы, предусматривая её реакцию на разнообразные внешние воздействия, в том числе память о результатах предыдущих воздействий по времени (конечный автомат).

Это единственное отличие компьютерных программ от других продуктов творческой деятельности человека является, тем не менее, принципиальным, так как до этого человеку крайне редко предоставлялась возможность

вложить в продукт своего труда алгоритм сложного поведения для самостоятельного взаимодействия его творения с окружающим миром (одним из исключений является разработка идеологий существования человеческого общества).

Таким образом, создание компьютерной программы можно с большей степенью приближения уподобить акту сотворения живого организма, чем рождение, например, новой научной теории или создание нового художественного фильма. И мы имеем право сделать такой вывод в силу того, что граница между «живым» и «неживым» определяется, в частности, и степенью сложности самостоятельного поведения «живого» и «неживого». Конечно, мы не станем утверждать, что компьютерная программа является «живой» в полном и привычном понимании этого слова, но степень ее приближения к этому состоянию и ее самостоятельная «жизнь» после «отчуждения» от автора (авторов) должна являться предметом внимательного разбора и изучения.

Компьютерная программа как живое существо

Начнём с того, что, кроме поведения, компьютерная программа обладает ещё одним свойством «живого» — способностью к размножению. Чаще всего этот процесс совершается опосредованно и при участии человека, который размножает объектный код программы при перезаписи её с одних носителей на другие, но в случае распространения исходных текстов программы возрастает и число её «мутаций» — видоизменений, выполненных другими программистами.

Существуют и саморазмножающиеся программы, хорошо известным примером которых является печально знаменитый класс программ–вирусов.

Большинство программ подвержены естественному отбору (при помощи человека), в котором лучшие программы заставляют «вымирать» более слабых конкурентов. При этом «выжившие» часто задают начальный уровень качества и определяют ключевые идеи, широко используемые в дальнейшем.

Многие программы обладают большими или меньшими способностями к самоорганизации (ещё одной особенностью, присущей живой материи). Они могут самостоятельно упорядочивать и переупорядочивать данные, которые накапливаются в их памяти за время взаимодействия с внешним миром и даже «отчуждать» от себя эти данные в виде отчётов, файлов, баз данных, а иногда — вместе с программами обработки этих данных — и в виде целых «организмов», способных к дальнейшему самостоятельному существованию.

Даже разработчики часто не могут детально и точно представить себе, что в данный момент происходит с их большой программной системой, и предсказать в точности её дальнейшее поведение. Более того, если в системе используются критерии вероятностного выбора поведения системы, эта задача может быть затруднена принципиально.

Правда, при необходимости разбора и исправления ошибок в таких системах разработчики специально предусматривают механизмы поиска и локализации ошибок, включающие автоматическую запись управляющих событий и реакций системы, программную трассировку, отладочные выдачи, «посмертные» дампы и другие средства, но при этом каждый, кто хоть раз отлаживал действительно большие программы, знает, насколько нелегко порой бывает разобраться в причинах ошибок и устранить их, не внося при этом новых ошибок. Особенно трудно это бывает в тех случаях, когда в системе широко используются параллельные процессы и/или от системы требуется работа в реальном масштабе времени.

Не зря в среде программистов бытует мудрость: «В каждой большой программе существуют как минимум три ошибки. После устранения любой из них в ней остаются или появляются ещё как минимум три. И этот процесс продолжается до самого окончания жизни программы».

Из всего этого видно, насколько сложной и похожей на «живой» объект может быть компьютерная программа, которая порождается разработчиками, и насколько может быть трудно, а порой просто невозможно предсказать её дальнейшую жизнь и поведение, особенно после «отчуждения» от создателя.

В этом месте я хотел бы проиллюстрировать вышесказанное примером длительной (десятилетней) разработки программ для большой информационной системы, базировавшейся на многомашинном комплексе. Участвуя в этом проекте в качестве разработчика, в том числе последние пять лет руководя всей разработкой и одновременно занимаясь практическим программированием, я детально представлял себе устройство всей системы в целом.

По мере возрастания числа выполняемых функций система росла и развивалась в заранее спланированных для неё рамках. Отдельные подсистемы, как это обычно бывает, проходили различные этапы приёмки, передавались сначала в опытную, а затем в промышленную эксплуатацию. Разработчиками осуществлялся авторский надзор и сопровождение, замеченные ошибки в программах устранялись, и хорошо отлаженные части системы постепенно

«отчуждались» от разработчиков. Они начинали взаимодействовать друг с другом, с персоналом, эксплуатировавшим систему, и с её пользователями, работавшими, в частности, с удалённых терминалов по линиям связи.

Надо сказать, что в период интенсивного авторского сопровождения программ я не задумывался над тем, «что же такое мы создаём?» с точки зрения его подобия «живому организму». Но когда стало ясно, что значительная часть системы уже «живёт» своей собственной, независимой от нас жизнью и авторы уже не могут существенно влиять на эту «жизнь», не разрушая систему, я испытал известное потрясение — продукт был «отчуждён» и жил самостоятельно! Причём, сохранность этой «жизни» гарантировалась заинтересованностью в нормальном функционировании системы множества других людей, пользующихся нашим «творением»... И чем больше проходило времени с момента «отчуждения», тем сильнее становилось ощущение самостоятельной «жизни» системы.

Думаю, что это чувство в той или иной степени знакомо почти всем программистам, продуктами труда которых пользуются другие люди. Подозреваю также, что это же чувство заставляет некоторых молодых программистов слишком гордиться своим умением написать небольшую программу и из–за этого несколько свысока относиться к окружающим.

Однако с опытом рано или поздно приходит понимание того, насколько превосходит любого из нас великий программист, сотворивший нашу Вселенную. Но вернёмся к компьютерным играм.

Компьютерная игра как вселенная

Действительно, создавая компьютерную игру, автор определяет общие законы, по которым существует игровая вселенная, разрабатывает объекты, подчиняющиеся этим законам, наделяет их поведением, проектирует их внешний вид, их трансформации и прочее. Иными словами, создаёт некий мир, в котором существуют и живут сконструированные автором создания.

Как уже говорилось, иногда этот мир может быть очень прост, жёстко детерминирован и неспособен к самостоятельному существованию и развитию, а иногда — весьма сложен, подчинён вероятностным законам и способен к продолжительной жизни даже без участия играющего.

Объекты, существующие в таком мире, могут быть сконструированы и как «живые» существа, обладающие сложным поведением и самостоятельно взаимодействующие друг с другом, наделены способностью к размножению и смерти, памятью, способностью к самообучению и многими другими свойствами.

И вновь хочу на примере собственного опыта показать, какими интересными свойствами может обладать «живая» игра.

Много лет назад на «больших» компьютерах, занимающих как правило целый машинный зал, был очень популярен класс игр под названием «Star Trek» («Звёздный Путь»). Игры с таким названием писались разными авторами и для разных компьютеров, но, в основном, под влиянием одного и того же одноимённого телевизионного сериала, пользовавшегося большой популярностью в те годы (как, впрочем, пользуется заслуженной популярностью этот же возрождённый космический сериал и в наши дни). Действие почти всегда происходило в далёких звёздных мирах, где космический корабль с большой командой на борту путешествовал среди звёзд, открывал новые миры, сражался с вражеским звёздными флотилиями, защищал свои Звёздные Базы, заправлялся энергией и ремонтировался, снова путешествовал и сражался и т.д.

Конечно, компьютерная графика в те времена была достаточно несовершенна и в лучшем случае предоставляла возможность работать с символами на алфавитно–цифровых дисплеях, но все равно «Star Trek» захватывал воображение, и многие из нас ностальгически вспоминают о нем до сих пор. А чего стоили «разборы полётов» с разгорячёнными после игры коллегами–программистами по дороге домой в метро! Соседи по вагону удивлялись, слыша что–нибудь вроде: «...и как только я вышел из гиперпространства...»

Романтика звёздных сражений настолько увлекала некоторых настоящих мужчин (программистов), что в определённый момент многие из нас начинали мечтать о своей собственной игре «Star Trek», которая была бы лучше уже существующей. Не всем удавалось воплотить эту мечту в реально работающую новую программу, но многие из нас с той поры оказались «заражёнными» компьютерными играми и до сих пор ищут на персональных компьютерах свой «игровой идеал» — современную версию старой игры «Star Trek».

Не устоял перед искушением написать свой собственный «Star Trek» в своё время и я — и он оказался моей первой игровой программой, на которой я ощутил всю необыкновенную прелесть такой работы.

Дело в том, что эту игру я разрабатывал в первую очередь для самого себя. Все, что мне приходило в голову и хотелось видеть в готовой игре, можно было попытаться запрограммировать. Один из постоянно доступных мне компьютеров был очень неплох — HP-3000. Памяти, по тем временам, было более чем достаточно, быстродействия —

тоже (к тому же я имел статус менеджера системы). Единственным ограничением, накладываемым на эту работу, являлось моё время, свободное от основной работы. Но программисту ли привыкать работать по ночам или съедать обед из термоса в выходные?

Одной из самых существенных концепций моей игры являлось вероятностное поведение объектов игры и всей компьютерной вселенной в целом. Это гарантировало невозможность точного предсказания дальнейшего развития событий в любой ситуации даже для автора, что и делало игру для меня наиболее привлекательной. Программа имела специальный генерационный файл данных, содержащий около 500 параметров, относящихся к различным объектам игры.

Звезды и планетные системы зарождались и умирали, подчиняясь сложным параметрическим законам с вероятностными критериями, сверхновые взрывались в определённых ситуациях и с предварительно заданной вероятностью, оборудование космического корабля и поведение его экипажа описывалось очень большим количеством параметров и, соответственно, могло оказаться совершенно различным в сходных ситуациях. Враги, обладая очень сложной и в целом детерминированной стратегией поведения, могли отклоняться от неё в рамках разрешённой вероятности их поведения. Каждый противник, таким образом, обладал своими склонностями и характером. Одни были агрессивны, другие — трусливы, третьи — склонны к компромиссам, переговорам и т.д. Если им удавалось договориться друг с другом, они могли в разной степени объединять свои усилия в борьбе со мной или наоборот — начать свою собственную внутреннюю войну. Враги рождались, жили и умирали по своим законам, нашу Галактику иногда посещали корабли из других галактик, команды которых могли присоединиться, в зависимости от обстоятельств, к той или иной враждующей стороне.

Наконец, на каком–то этапе разработки, игру стало возможно предоставлять самой себе на длительный промежуток времени, например, на ночь, и утром узнавать, к чему привело развитие событий в отсутствие автора. Если появлялось желание, можно было принять участие в игре или оставаться просто наблюдателем. (На нашем HP-3000 игра без участия человека выполнялась как задача с невысоким приоритетом, никому особенно не мешая). Примерно в это же время я решил добавить в игру возможность одновременной работы с нескольких терминалов для разных участников; и на этом этапе (примерно через год после начала работы) разработка игры, собственно, закончилась. Настало время наслаждаться своим «творением» и иногда отдыхать за дисплеем компьютера, а также можно было подвести некоторые итоги.

Но именно в этот момент у меня появился мой первый персональный компьютер Yamaha MSX, обладавший цветной графикой и захвативший на длительное время все мои помыслы.

Итоги разработки большой игры

Итоги разработки этой большой игры были подведены мной значительно позже и, честно говоря, обрадовали и даже немного испугали одновременно. Я был рад, что написал такую большую и сложную программу и выполнил задачу, которую перед собой поставил в полном объёме. Я гордился тем, что в очередной раз создал систему, способную к дальнейшему существованию без участия автора, и что моей игровой программой с удовольствием пользовались некоторые мои коллеги по работе. Но одновременно с этим я понял также и то, что на этот раз соприкоснулся с чем–то большим, чем обычное проектирование компьютерной программы. И осмысливая этот неожиданный опыт, мне пришлось сделать некоторые выводы, не имеющие прямого отношения к программированию в общепринятом понимании этого слова.

Во–первых, мне довелось на практике убедиться в справедливости концепции, носящей название «Лучший из миров». Суть этой философской и физической концепции, как известно, заключается в том, что мы живём в «лучшем из миров» в том смысле, что наша Вселенная является устойчивой в силу вполне определённого соотношения значений таких физических констант, как скорость света в пустоте, постоянная Планка, заряд электрона и др. Соответствующими теоретическими расчётами показано, что существует лишь очень небольшая область значений этих мировых констант, обеспечивающая устойчивость существования материи. И наша Вселенная принадлежит к этому множеству. В случае даже небольших отклонений значений этих констант в ту или иную сторону, материя начинает стремительно дезорганизовываться и превращается в хаос или, что то же самое, не может организоваться из первичного хаоса вовсе.

Как я уже говорил, моя игра имела около пятисот параметров, определяющих её игровую вселенную, и правила поведения объектов, включая вероятностные характеристики отклонений параметров от их стандартных значений. В процессе проектирования и настройки игры с помощью этих параметров выяснилось, что возможные устойчивые игровые вселенные также занимают очень небольшой объем в многомерной области допустимых значений всех параметров. А при выходе одного или нескольких параметров за границы этого объёма игровая вселенная начинает стремительно деградировать и становится однородной и неинтересной для играющего. Например, сверхновые

звезды вспыхивают слишком часто и сжигают все вокруг себя, новые звёздные системы зарождаются слишком редко или слишком часто, персонажи самоуничтожаются, вымирают или уничтожают друг друга в братоубийственных войнах, оборудование корабля становится слишком ненадёжно, команда корабля постоянно бунтует и плохо выполняет приказания или, наоборот, проявляет излишнюю самостоятельность и делает то, что делать совсем не надо... Перечисление можно легко продолжить. Таким образом, понятие «Лучший из миров» для данной игры приобрело сугубо практическое значение, и мне пришлось выбирать этот «Лучший из миров» аналитически, интуитивно, а также методом проб и ошибок в процессе настройки программы. Более того, стало особенно важно попасть в один из «Самых лучших миров» для того, чтобы игровая вселенная могла самостоятельно существовать без моего участия.

Кроме того, постепенно развивая игру и наделяя её персонажей характерами с различными вероятностными характеристиками, я почему–то не задумывался о том, что, вообще–то, ничто не мешало мне запрограммировать у них интерес к познанию окружающего их мира и даже способность к рефлексии. Те «существа», которых я создавал, уже обладали некоторой, независимой от меня индивидуальностью. Они могли рождаться и умирать без моего вмешательства, имели свои собственные имена, которых я не знал до тех пор, пока не знакомился с каждым из них по отдельности, обладали достаточно сложным поведением, в большей или меньшей степени адекватным окружающему их миру, некоторым небольшим запасом слов и понятий (я с ними беседовал и иногда мог даже «убедить» в чем–то), они объединялись в группы с общими интересами или, наоборот, враждовали друг с другом. В моей модели не хватало совсем немногого: их собственной заинтересованности в продолжении игры и осмысления ими в этой игре своей роли. Кроме того, хотя они и «догадывались» о моем существовании, но не могли целенаправленно задавать мне вопросы о том мире, в котором существуют, и получать от меня соответствующие ответы.

Когда я впервые задумался над этим, я уже, к сожалению, не имел доступа к тому компьютеру, на котором создавал эту игру. Поэтому я не имел и возможности попытаться дополнительно запрограммировать эти черты их «личности» и посмотреть, что же из этого получится. Однако, чисто теоретически, эта задача не представляется мне особенно сложной, так как ещё с очень давних времён у меня уже был опыт разработки обучаемых программ, синтезирующих связный русский текст с расширяемым и процессе обучения запасом понятий и оперирующих различными категориями (в том числе даже морально–этическими). А развитый язык понятий — один из основных критериев степени развития интеллекта и знаний об окружающем мире!

Дальнейшие размышления по поводу того, во что могла бы превратиться моя игра, привели меня к некоторым неожиданным религиозно–философским аналогиям. Я задавал себе вопросы и пытался отвечать на них. Итак:

- Зачем я создавал эту программу? — Отчасти ради самого удовольствия от процесса её создания и предвкушения удовольствия от дальнейшей «игры» с ней. Отчасти для расширения разнообразия окружающего меня мира. Отчасти для удовлетворения чувства созидания, живущего в каждом человеке. Отчасти просто из любопытства. И ещё мне хотелось иметь возможность исследовать в деталях все время новый, постоянно обновляющийся мир игры, сталкиваясь порой с неожиданными явлениями в рамках предварительно определённых мной законов этого мира.
- Почему я вынужден был искать для этой игры свой «Лучший из миров»? — Потому, что вырожденная игра, хаос мне неинтересны. Мне нравится уменьшать энтропию, идти от хаоса к упорядоченности.
- Зачем «живут» мои персонажи? — Для того, чтобы, изменяя в бесконечных сочетаниях и вариациях свой мир, доставлять мне постоянную радость исследования и познания этого изменяющегося мира.
- Как я отношусь к самоуничтожению своих персонажей? — Резко отрицательно. Любое «самоубийство» персонажа, с моей точки зрения, есть невыполнение им возложенных мной на него надежд на участие в игре.
- Как я отношусь к персонажам своей игры? Ведь среди них есть «плохие» и «хорошие», «добрые» и «злые», «умные» и «глупые»... — Конечно, субъективно, как участнику игры, мне приятней иметь дело с «хорошими», «добрыми» и «умными» персонажами, но без «плохих» игра выродилась бы и стала бы мне неинтересна. Объективно я могу сказать, что мне необходимы все персонажи и что я, как творец, одинаково люблю их всех, хотя, безусловно, больше сочувствую «хорошим». Необходимость во всех персонажах становится особенно заметной в том случае, когда я лично не участвую в игре и выступаю лишь в качестве наблюдателя.
- Как поведёт себя мой персонаж, если в результате рефлексии вдруг придёт к выводу, что он является «существом», а я являюсь творцом его мира? — Предположительно, он захочет задать мне вопросы и получить на них ответы. Но большая часть моих ответов в той или иной степени будет соотноситься с моим миром, о котором этот персонаж не имеет никакого представления. Какая бездна новых знаний и понятий! Захочет ли он после этого возвращаться в свой простой игровой мир или будет все время стремиться «медитировать», общаясь со мной? Интересно и то, что подавляющая часть часть знаний, которые я мог бы ему сообщить, оказалась бы совершенно неприменимой в рамках его вселенной.
- Как поступать с машинной памятью и хранящимися в ней упорядоченными за время «жизни» персонажа данными, остающимися после его «смерти»? — В этом месте каждый программист, читающий эту статью, может сам поразмыслить на эту тему и предложить наиболее соответствующее его пониманию проблемы и вкусам решение.

Здесь можно было бы поставить точку. Но просится многозначительное многоточие...

Могу добавить, что разработка и анализ этой игры в своё время так сильно повлияли на моё мировоззрение, как не смогли повлиять до этого никакие теоретические и религиозные концепции. Согласитесь, одно дело — рассуждать о чем–то схоластически, а другое дело — попробовать самому.

О технологии разработки игровых программ

Теперь от «большой игры» на старой «большой машине» перейдём к рассмотрению технологий разработки игр для современных персональных компьютеров.

Как известно, процесс создания таких игр состоит из четырёх составляющих:

- написание сценария;
- разработка графики игры;
- написание музыкального сопровождения;
- разработка компьютерной программы, объединяющей все три вышеперечисленные компоненты в единое целое, называемое компьютерной игрой.

Для того чтобы процесс создания игры средней сложности не растянулся на несколько лет и конечный продукт был бы получен в короткие (или просто разумные) сроки, необходимо иметь определённый набор инструментов, который я в дальнейшем для краткости буду называть просто «верстак».

Если таких инструментов нет или их не хватает, то, прежде чем приступить к разработке игры, лучше всего создать эти инструменты самому. Ведь вы не станете, к примеру, делать табуретку, если у вас нет рубанка! Конечно, можно попытаться выстругать детали табуретки перочинным ножом или топором, но какая табуретка при этом получится и сколько времени это займёт? А сколько новых табуреток можно было бы изготовить за то время, пока вы делали эту?

Итак, что же должно быть на вашем «верстаке»?

1. Удобный редактор текстов для написания сценария, подготовки документации и разработки исходных текстов программ.
2. Мощный графический редактор для подготовки фоновых задних планов и анимационных объектов. Если планируется использование трёхмерной графики, то потребуется и 3D–редактор. Неплохо также иметь сканер, видеокамеру и устройство ввода в компьютер видеоинформации. Форматы хранения изображений в файлах на дисках должны поддерживаться удобными и простыми библиотечными функциями для последующей загрузки графики в вашу программу. Если вы планируете использовать в игре видео, может быть, вам придётся снимать и монтировать целые фрагменты видеофильмов, чтобы потом показывать их с помощью вашей программы.
3. Удобный музыкальный редактор, работающий с той музыкальной аппаратурой, которую вы рассчитываете использовать в своей игре. Так же, как и для графики, вам должны быть доступны соответствующие библиотечные функции для удобного проигрывания созданной музыки из вашей программы.
   Хорошо, когда музыка, используемая в игре, мелодична и удачно аранжирована. Поэтому, если вы не уверены в собственных силах, попросите написать музыку для вашей игры профессионального музыканта. В противном случае ваша игра может потерять привлекательность даже при интересном сценарии и хорошо продуманном алгоритме. (То же самое, впрочем, относится и к графике.)
4. Необходим компилятор с языка программирования, на котором вам удобнее всего излагать свои мысли, и, может быть, ассемблер. С точки зрения компактности письма и обозримости текста программы лучше всего писать и отлаживать свои программы на языках высокого уровня, но, если эффективность порождаемого компилятором кода оказывается недостаточной, отдельные (наиболее критичные по быстродействию или использованию памяти) функции придётся переписывать на ассемблере.
   Для себя, например, я сделал выбор языка программирования несколько лет назад, остановившись на языке Си. Перед этим я писал программы на более чем десятке разных языков и ассемблеров на различных компьютерах, сам разработал и реализовал несколько специализированных языков и макропроцессоров, писал даже непосредственно в двоичных кодах и абсолютных адресах (первый раз ещё в школе), но, познакомившись с Си, понял, что в терминах этого языка мне удобнее всего думать и излагать свои мысли.
5. Вам могут потребоваться отдельные утилиты, осуществляющие преобразование форматов данных, осуществляющих защиту от копирования ваших программ, средства работы с библиотеками и автодокументатор для исходных текстов или какие–нибудь другие средства в зависимости от необходимости и конкретных потребностей каждого разработчика.
   В случае коллективной разработки проекта, вам потребуются соответствующие средства коллективной

разработки и сопровождения комплекса программ, а также команда бета–тестеров (альфа–тестерами выступают сами разработчики).
6. И, наконец, может быть, самое важное. Вам нужна библиотека функций, с помощью которой можно легко и удобно работать с графическими и звуковыми данными, опрашивать все необходимые вводные устройства (мышь, джойстик, клавиатуру, специальные контроллеры), осуществлять анимацию объектов и управлять ими, следуя логике вашей игры и не отвлекаясь на «заклинания», необходимые для взаимодействия с компьютером через различные системные интерфейсы, требующие выполнения множества системных соглашений.

Проблема «заклинаний» иногда приобретает просто чудовищный характер в некоторых мультизадачных средах, где большинство системных ресурсов, к которым необходимо получить доступ, требуется сначала описать, потом открыть, заполнив предварительно подготовленные соответствующие структуры данных, и, наконец, поработав с этими ресурсами (далеко не всегда удобными для вас лично примитивами системных операций), освободить, закрыть и вернуть системе.

Если вы не будете делать всего этого, вам скорее всего придётся довольно часто перезагружать систему, а ваша новая программа, с большой степенью вероятности, не будет работать на других компьютерах. Порой, пока вы планируете и программируете доступ к этим ресурсам, можно вообще забыть, зачем вы сели за компьютер и какую задачу собирались решать. (Достаточно удобным и корректным выходом из этой ситуации является погружение всех «заклинаний» в ваши собственные библиотечные функции и макросы при условии, конечно, что они понадобятся вам и в дальнейшем.)

Но это лишь одна из причин, по которой разработку каждого нового класса программ лучше всего начинать с проектирования или расширения вашей собственной библиотеки.

Другая (и более важная) причина заключается в том, что, если вы хотите действительно быстро писать программы данного класса (в частности, игры), не отвлекаясь на второстепенные детали, вам необходимо определить, в каких терминах и операциях вам удобнее всего описывать алгоритмы работы программ этого класса (включая языковые конструкции используемого вами языка программирования). Другими словами, следует понять, каким вам хотелось бы видеть язык, на котором вы будете писать свою программу. Это очень важная работа и её желательно проделать в начале, т.е. перед тем, как написать самую первую строку вашей самой первой программы любого нового для вас класса. (В этом могут оказать большую помощь объектно–ориентированные языки программирования).

Результатом этой работы должно стать множество структур данных и операций, минимизированное по количеству функций и их параметров, что позволит наиболее компактно описать алгоритм работы вашей программы. Это очень тонкая работа, требующая известного опыта и являющаяся своего рода искусством: исходя из общей постановки задачи (например, в виде сценария) и знания архитектуры вашего компьютера (или архитектур всех известных вам компьютеров), следует придумать максимально эффективные примитивы операций, с помощью которых вы могли бы предельно компактно описать все ваши алгоритмы и которые не снижали бы существенно эффективности работы ваших готовых программ!

Сверхзадачей при этом является максимально возможная мобильность (переносимость на компьютеры с другой архитектурой) ваших программ за счёт того, что вся конкретная работа с аппаратурой каждого отдельно взятого компьютера может быть полностью описана вашими новыми примитивами. Таким образом, один раз переписав ваши примитивы для компьютера с другой архитектурой и пользуясь родственным компилятором того же языка программирования, вы сможете в дальнейшем легко переносить ваши программы данного класса на другие компьютеры.

При внимательном рассмотрении данной концепции легко заметить, что множество примитивов, о которых шла речь выше, вместе с конструкциями используемого языка программирования образуют некоторый новый язык, на котором и описывается алгоритм работы программы. Часть примитивов погружается в библиотеки, а часть выносится на уровень макросов языка программирования. При этом, как показывает практика, на уровень макросов чаще всего выносятся узкоспециализированные обращения к тем же библиотечным функциям или их использование в различных частных комбинациях, увеличивающие выразительную силу нового языка.

Правда, такой подход несколько противоречит концепции структурного программирования Н. Вирта, в которой любая разработка должна проектироваться догматически строго сверху вниз, но, во–первых, Вирт и его последователи замалчивали этап предварительной разработки языка под задачу, во–вторых, в программах, работающих в реальном масштабе времени, всегда важно знать и оценивать реакции аппаратуры на различные сложные воздействия, а не опробовав каждую новую программную функцию нижнего уровня, такие оценки давать очень трудно. К тому же разработка большинства программ идёт почти всегда сразу в двух направлениях: от аппаратуры — снизу вверх и от логики разработки всего проекта — сверху вниз. Интересно, что после «обкатки» каждого нового набора примитивов на нескольких программах данного класса становится возможным и даже удобным пользоваться в дальнейшем исключительно структурным подходом при проектировании новых программ

данного класса.

Таким образом, ваш «верстак» должен содержать немалое количество хороших специальных инструментов, подходящих для такой работы. Но если вы не пожалеете времени на сбор этих инструментов, их освоение и разработку недостающих, то скоро у вас появится возможность «печь» наши программы как блины — быстро и качественно!

На своём опыте я уже не раз убеждался в этом. Потратив известное время на изготовление «верстака» для совершенно «голого» в те времена компьютера MSX (1986–1987 гг.), я затем всего за полгода написал четыре(!) игровых программы выше средней степени сложности. Причём каждая из них разрабатывалась с нуля, включая сценарий, музыку, графику и собственно игровую программу. Таким образом, на каждую игру от начала разработки (первого слова сценария) до её окончательного завершения (вместе с музыкой и графикой) уходило всего полтора месяца работы одного человека!

Все четыре игры относились к классу «приключенческих» игр (Adventure), обладали сложными сценариями, развитой анимацией, экранными скроллингами больших полиэкранов, специальной многооконной организацией графической информации и генерировали свои игровые вселенные с учётом вероятностного распределения параметров, что исключало возможность повторения одних и тех же игровых ситуаций. Каждый сеанс любой из этих игр был новым приключением с новыми отличиями в каждой конкретной игре.

Этот подход я использую и сейчас, в том числе при разработке программ для компьютеров с другой архитектурой. Помимо всего прочего, хороший «верстак» всегда предохранит вас от множества мелких и досадных ошибок, так как позволит максимально компактно описывать алгоритм работы вашей программы.

Но, углубившись в пучину технологий, не забывайте, пожалуйста, что игры бывают «добрые» и «злые», а также, что, согласно Словарю русского языка (под ред. А.П. Евгеньевой), одно из определений игры — «свойственное некоторым винам и шипучим напиткам движение пузырьков газа».

Заключение

Я надеюсь, что если хотя бы один читатель этой статьи узнает из неё что–нибудь новое или полезное для себя лично, то моя задача уже выполнена.

А если какой–нибудь мой коллега–программист прольёт слезу умиления над этой статьёй, потому что сам давным–давно пришёл к тем же выводам, я буду просто счастлив.

Во всех остальных случаях не судите меня слишком строго, ведь игры — дело совершенно несерьёзное. В них играют маленькие дети. А большие дети их разрабатывают.

(и не забывайте, пожалуйста, что эта статья написана в 1993 г. Многое с тех пор изменилось, но основы, во многом, остались теми же)

* * *

Оригинал

http://sysadminmosaic.ru/articles/rodionov_myowngame

2024-03-07 11:15