

Когда нужно создавать динамические веб-сайты, то PHP гораздо быстрее, легче и проще изучить чем альтернативные языки. При этом вы еще получаете прекрасное быстродействие и интеграцию практически с любой известной на сегодняшний день базой данных, а также стабильность и бесконечное количество доступных при настройке опций (PHP расширяемый язык).

ЛИТЕРАТУРА

1. «Создаем динамически веб-сайты с помощью PHP, MySQL и JavaScript» Робин Никсон 2012, Издательство: Питер.
2. Диск "Все Технические Моменты Онлайн Бизнеса в Видеоформате 2 или PHP+MySQL для начинающих" Евгений Попов, 2006
3. <http://softlab-portable.ru/index.php?newsid=147176>. Sothink Logo Maker
4. <http://tvoyuspex.ru/multimedia-2/writing-and-editing/kak-zapisyvay-videouroki-v-camtasia-studio-7/>. Как записывать классные видеоролики в Camtasia Studio 7 А
5. <http://vsfte.biz/220-camtasia-studio-7.html>. Camtasia studio 7 – мощная программа для создания и монтажа видео.

УДК 004.5(075.8)

Студ. А.П. Горбач

Науч. рук. ст. преп. Т.В. Кишкурно
(кафедра информатики и веб-дизайна, БГТУ)

ИСПОЛЬЗОВАНИЕ ПРОТОТИПОВ ПРИ ПРОЕКТИРОВАНИИ ИНТЕРФЕЙСОВ

Прототипирование ПО (программного обеспечения) становится всё более популярным и часто используемым процессом в белорусских IT-компаниях. Причины видятся следующие: с одной стороны – это определенная дань моде, с другой – прототипирование обещает компании ряд весомых преимуществ.

Прототипирование – быстрая «черновая» реализация базовой функциональности для анализа работы системы в целом. Прототип позволяет также получить обратную связь от будущих пользователей, причем, именно тогда, когда это наиболее необходимо: в начале проекта еще есть возможность исправить ошибки проектирования практически без потерь.

Однако сделать процесс прототипирования полезным и эффективным – непростая задача. Встречаются подводные камни, появляются вопросы. Кто и когда должен прототипировать? Как делать про-

тотипы? Как их использовать? Ответы на эти вопросы и последующие шаги определяют успешность и полезность нововведения.

Согласно международным стандартам в этой области, таких как SWEBOOK (Software Engineering Body of Knowledge – свод знаний по программной инженерии), IEEE 830-1998 (рекомендации IEEE по разработке требований к программному обеспечению) прототипы – это отличный инструмент для уточнения и/или детализации требований. Они полезны по следующим причинам: 1) потребитель может предпочесть посмотреть и оценить прототип, нежели читать и оценивать SRS (Software Requirements Specification – спецификация требований программного обеспечения), поэтому прототип обеспечивает быструю обратную связь; 2) прототип демонстрирует неожиданные аспекты поведения системы – это помогает более полно проанализировать SRS; 3) SRS, основанная на прототипе, имеет тенденцию меньше подвергаться изменениям во время разработки, тем самым сокращая время разработки.

Таким образом, формулируется первая цель прототипирования – решение проблемы недопонимания между аналитиком и пользователем.

Исходя из этого, можно выделить следующие варианты использования прототипов:

- как инструмент извлечения, проверки и утверждения требований на этапе работы с требованиями;
- как основу для написания SRS и ТЗ (технического задания) на этапе проектирования;
- как технику проверки программного дизайна на этапе проектирования;
- как объект исследования юзабилити-тестирования на этапе тестирования;
- как образец для разработчиков на этапе реализации (конструирования).

Однако прототипы можно использовать ещё более широко:

1. На этапе коммерческого предложения. Прототип позволяет выделиться из многих похожих коммерческих предложений и расположить к себе заказчика.

2. Как образец при тестировании готового ПО. Некоторые интерфейсные функции тяжело описываются и воспринимаются в текстовом виде, и в этом случае прототип является хорошим помощником.

3. Как образец при приёмке-сдаче работ. Когда заказчик вместо проверки функций по ТЗ, сравнивают реализованную систему с прототипом. Это выгодно обеим сторонам. Заказчик вправе предъявить

претензии, если в реализованной системе что-то не так, как в прототипе. Но и исполнители могут защитить себя от претензий, если реализуем систему точно так же, как в прототипе.

4. Как пример решения для демонстрации потенциальным заказчикам. Иногда нет возможности продемонстрировать заказчику готовую систему, тогда ему показывают прототипы, разработанные и использованные на прошлых проектах.

Жизненный цикл прототипа состоит из следующих этапов: 1). Создание и демонстрация прототипа на этапе проектирования; 2). Итерационная разработка и согласование прототипа; 3). Проверка удобства использования прототипа; 4). Проверка прототипа на реализуемость; 5). Создание ТЗ на основе прототипа; 6). Реализация ПО с использованием прототипа в качестве образца; 7). Тестирование ПО с использованием прототипа в качестве образца; 8). Проверка ПО заказчиком на соответствие прототипу; 9). Доработка прототипа для новых требований.

Как видно из жизненного цикла, прототип пронизывает весь процесс разработки. Прототип стал элементом, который все видят и который все обсуждают: от пользователя до программиста. Он стал своего рода объединяющим, центральным звеном. Если раньше процесс передачи информации выглядел примерно так (рис. 1):

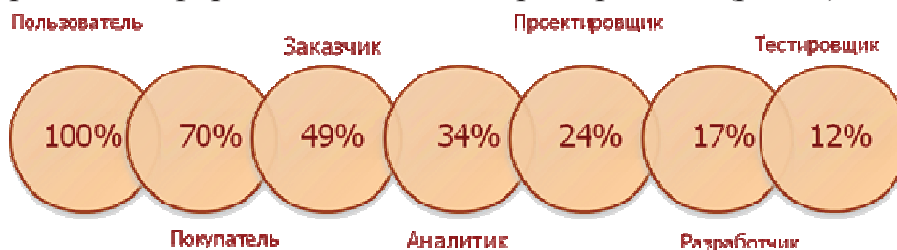


Рисунок 1

То сейчас он представляет собой что-то вроде этого (рис. 2):



Рисунок 2

Таким образом, благодаря прототипу все «звенья» близко связаны и находятся в непрерывном взаимодействии, что значительно об-

легчает разработку, уменьшает количество ошибок и помогает лучше распределить ресурсы проекта.

Внедрение прототипа в дизайн позволяет более точно воплотить в жизнь то чего именно хочет от нас заказчик. Простота создания, оперативность внесения изменений, интерактивность, эстетичность внешнего вида и даже частичное тестирование юзабилити – основные преимущества создания прототипа. При разработке прототипа заказчик видит пример и соглашается или не соглашается с направлением и дизайном, которые будут реализованы на его проекте, или указывает какие-то замечания или пожелания. Такая система активного использования прототипа и постоянного общения с заказчиком позволяет значительно уменьшить число конфликтных ситуаций, сократить число затраченных ресурсов и времени. Таким образом можно сделать вывод о том, что прототипирование в дизайне это отличное средство сэкономить ресурсы, время, а добиться хорошего результата при разработке проекта.

Из всего вышперечисленного можно сделать вывод о том, что прототипирование необходимо сегодня для успешной реализации проекта. Это выгодное средство достижения лучшего и качественного результата.

УДК 004.921

Студ. А.А. Гребенчук

Науч. рук. ассист. И. А. Миронов

(кафедра информатики и веб-дизайна, БГТУ)

СОЗДАНИЕ ИНТЕРАКТИВНЫХ 3D МОДЕЛЕЙ В СРЕДЕ UNITY

С развитием GameDevelopment'a требования к игровому контенту, в частности создания 3D игр, и программам, которые их создают, увеличиваются с каждым месяцем, задавая более высокие темпы развития игровой индустрии. На данный момент существует много платформ для создания различного игрового, обучающего контента, но некоторые из них не доступны рядовому пользователю, так как предоставляются только в платном варианте, а некоторые сложны для пользователя не знакомого с программированием. Но существуют и исключения, такие как Unity 3D, Torque 2D/3D, CryEngine 3, UDK (Unreal Development Kit).

Наш выбор пал на Unity 3D из-за некоторых его достоинств по сравнению с другими платформами:

- Unity – бесплатный кроссплатформенный 3D-движок;