Arcade Game Maker
Product Line –
Production Plan

ArcadeGame Team

July 2003
# Table of Contents

1 **Overview** 1
   1.1 Identification 1
   1.2 Document Map 1
   1.3 Concepts 2
   1.4 Readership 2

2 **Strategic view of product development** 3
   2.1 Assumptions 3
   2.2 Qualities 3
      2.2.1 Product qualities 3
      2.2.2 Production process qualities 3
   2.3 Products possible from available assets 4
   2.4 Production strategy 4

3 **Overview of available core assets** 5
   3.1 Test cases 5
      3.1.1 Unit tests 6
      3.1.2 Integration tests 6
      3.1.3 System tests 6
   3.2 Basic inputs and dependencies 6
      3.2.1 Inputs 6
      3.2.2 Dependencies 6
   3.3 Variations 7
      3.3.1 Absorbing vs reflecting 7
      3.3.2 Event Handling 7

4 **Detailed production process** 8
   4.1 Outline 8
   4.2 The process 8
      4.2.1 Product identification, definition and analysis 8
      4.2.2 Design new game 8
      4.2.3 Build Code 9
      4.2.4 Test the New Game 9
5  Tailoring production plan to product-specific production plan 11

6  Management information 12
   6.1 Schedule 12
   6.2 Production Resources 13
   6.3 Bill of materials (BOM) 13
   6.4 Product-specific details 14
   6.5 Metrics 14
      6.5.1 New lines of code 14
      6.5.2 Unique lines of code 14

7  Changing the Production Plan 15

8  References and Further Reading 18
1 Overview

1.1 Identification
The Arcade Game Maker Product Line organization will produce a series of arcade games ranging from low obstacle count to high with a range of interaction effects. More detailed information can be found in the Arcade Game Maker scope document. This document describes how a product is produced in the AGM product line.

1.2 Document Map
The Arcade Game Maker Product Line is described in a series of documents. These documents are related to each other as shown in Figure 1. This map shows the order in which the documents should be read for the first time. Once the reader is familiar with the documents, the reader can go directly to the information needed.

This is the production plan. Product Line organizations use this document to capture how the product teams will build a new product. This document follows the outline provided in [Chastek 02].

1.3 Using this document
This document is a template for product-specific production plans. Chapter 6 is the site of the major items that must be modified to make this the production plan for a specific product. The schedule and Bills of Material should be made specific to the product.
Figure 1 - Document Map

1.4 Concepts

See the Glossary document for definitions of basic concepts.

1.5 Readership

This document is intended to provide some level of information to all of the stakeholders in the Arcade Game Maker framework but is primarily intended for product development teams. The production plan describes for a manager the resources that are needed to produce a product. Technical members of the organization can use the production plan as the detailed process for producing a product.
2 Strategic view of product development

2.1 Assumptions

?? The company has an existing pool of software developers who are highly technical. They have fielded products on a variety of hardware platforms and are accustomed to being involved down to the driver level of the software.

?? The product line contains games that are similar in content but that differ in platform. Differences in platform translate into differences in graphics implementation, which is a major feature of these products.

2.2 Qualities

We will briefly discuss two types of properties: product qualities and production process qualities.

2.2.1 Product qualities

The products must be enjoyable to play in order to be a success. This requires both a colorful display and realistic action.

?? Any new game elements should add to the quality of the display. It should be colorful and representative of the item it represents.

?? The action of the game must proceed sufficiently fast to demand the player’s attention. When constructing a game, if the number of elements slows the game, alternatives must be investigated.

?? The action of the game must look like what the player expects. The motion and reactions of movable elements must be realistic. As elements are added to the game, their actions and their boundaries must be correctly set through parameters so that collisions appear real.

2.2.2 Production process qualities

The production process in the AGM product line is largely manual. The personnel are very technical so the process allows for hands-on manipulation of the product. To provide these technical personnel with a tool that automates the production process to a high degree would
have been a waste of their talent, frustrating for them, and a poor use of the resources needed to produce the automation.

2.3 Products possible from available assets

The products that are possible from the available assets only include games like those constructed so far. Using the incremental approach the asset base is only developed to the extent needed to construct the current set of products. As additional increments are completed, the variety of games that are possible will greatly increase.

2.4 Production strategy

The production strategy for the AGM product line includes a domain-based design approach and a manual construction approach. The analysis of requirements and development of the architecture will be based on domain information. Each new product will be built by having software developers manually specialize assets where necessary, gather other domain-based, core assets that are needed and then build an executable by running a compiler and linker.

The core assets are being built incrementally. The product teams for the earlier products will have fewer assets to choose from than will the teams on later products. For earlier product teams, one responsibility will be to identify candidates for core assets in the later increments. The freeware development will provide many opportunities for this.
3 Overview of available core assets

The available core assets include analysis, design, and implementation assets.

3.1 Naming conventions
Names that end in “Interface” are interface definitions.

Names that start with “test” are DotUnit test case suites.

3.2 Analysis-level assets
The requirements document includes several useful assets.

?? The domain analysis model organizes the concepts in the main domain, games, and provides the attributes of each concept as well as the relationships among them. Developers who are new to the domain should study this asset to understand the background for the product on which they are working.

?? The feature model shows the features of products in the product line.

?? The use case model provides a superset of requirements for products. Select the appropriate use cases for your product.

3.3 High-level design assets
The main high-level design asset is the software architecture. The architecture is described in detail in the two-volume architecture description.

3.4 Source code
Each of the interfaces in the existing architecture has been implemented. The C# components are named according to the names given in the architecture documentation.

3.5 Test cases
Test cases may be reused in building a product when the tests cover functionality or features that are included in the new product. This section describes existing test cases.

Test cases are not available for all assets yet. Here we describe the current status of test assets.
3.5.1 Unit tests
Individual unit tests are constructed using the DotNet testing framework. The source code is in the form of classes.

Unit test classes are currently available for:
   ?? Velocity/Speed/Direction cluster
   ??

3.5.2 Integration tests
Integration of units that result in a component is tested at the API level. Integration of units that result in GUI level is tested using the same procedure as the system test.

The current integration tests that are available are:

3.5.3 System tests
System tests are currently administered by hand. Each system test is a scenario that is derived from a specific use case. When a use case is used on more than one product, the related test cases can also be used on that new product. These test cases are documented in the test plan for a product.

Test plans are currently available for:
   ?? Brickles
   ??

3.6 Basic inputs and dependencies
3.6.1 Inputs
   ?? The main inputs to each game are mouse and keyboard events.

   ?? When the change case for saving a game is implemented, an additional set of inputs will be the data saved to the file.

3.6.2 Dependencies
   ?? The game products are heavily dependent upon the graphics library of the programming language.
When the change case for saving a game is implemented, the game products will be dependent on the operating system.

3.7 Variations

3.7.1 Absorbing vs reflecting

The stationary game elements participate in the game by providing behavior during a collision. There are two major behaviors available in the core assets at the moment. A stationary element may reflect the movable element according to the laws of physics or the element may absorb the movable element and cause it to be deleted from the game. A parameter on each element determines which of these behaviors is performed.

3.7.2 Event Handling

The event handling routines vary from one game to another. An implementation of the EventHandlerDefinitions interface is provided as a parameter to the GameBoard component. Each of the mouse and keyboard events are handled in a way defined in this implementation.
4 Detailed production process

This step-by-step process was developed as the AGM developers built the Brickles and Pong freeware games. It will be used by future product teams.

4.1 Outline
The process for producing a new product has five major steps:

?? Product definition and identification

?? Incremental analysis

?? Design product

?? Build product

?? Test product

4.2 The process

4.2.1 Product identification, definition and analysis
1. The products were identified as part of the product line planning process. Since each product is a single game, identifying the game to be implemented identifies the product.

2. The definition of the game consists of defining the rules of the game. There are several versions of most of these games so the product team must first decide on a set of rules to implement.

3. Analyze features for the new game that are variations from previous games; Identify existing features that must change for this game

4.2.2 Design new game
1. Plan how to provide those features from existing components

2. Plan how to provide the remaining features from new assets

3. Design the new implementation of the EventHandlerDefinitions interface.
4.2.3 Build Code
1. Start new ClassLibrary in a Visual Studio Project using {game name}Definitions as its name – Use this for new classes other than the game definition itself
2. Start new Windows Application in a Visual Studio Project using the name of the game.
3. Copy the Form1.cs file from a previous product line project
4. Change the namespace name to the new game
5. Configure the new gameboard
6. Copy data.txt from a previous game’s working directory. This is the resource file for the game.
7. Edit data.txt to reflect the new game
8. Compile the resource file
9. Copy the compiled resource file to the Debug directory

4.2.4 Test the New Game
1. Each core asset is tested, either by inspection or execution, as it is created or revised. When the asset is revised, the previous testing materials are revised and reapplied to the new version of the asset.
   
   For a code asset, the unit test asset is coded as a DotUnit test class.
   
2. The initial generic game test set is revised for each new game. In addition a game specific test set is created.

![Diagram of test cases related to products](image)

*Figure 2 - Test cases related to products*
3. The system test cases are maintained as text documents and are applied by hand.
5 Tailoring production plan to product-specific production plan

The production plan is, in general, very generic. The information in the plan applies to all products built using the current asset base. There are, however, some sections of the plan that must be modified for a specific product. Here we are giving a guide to product teams using this plan on how to modify it for their specific product.

The two most important and obvious sections that must be modified are the schedule and the bill of materials. Since we are using a manual product production approach the schedule defines which personnel are needed and when. The bill of materials will provide a useful means of tracking the use of core assets.
6 Management information

6.1 Schedule

In the product specific production plan this section contains the detailed schedule based on the process defined in section 4. Here we provide time estimates for most tasks. In the product-specific production plan this template is completed. The personnel assignments are made to specific people and the time estimates are changed based on any special circumstances for the product.

Table 1 – Time Estimates for producing the Brickles product

<table>
<thead>
<tr>
<th>Process step</th>
<th>Product specific</th>
<th>Who</th>
<th>When</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Identification</td>
<td>Product specific</td>
<td>Product planning</td>
<td>Done</td>
</tr>
<tr>
<td>Product Definition</td>
<td></td>
<td>analyst</td>
<td>.5 day</td>
</tr>
<tr>
<td>Product Analysis</td>
<td>Analyze the rules of the new game</td>
<td>analyst</td>
<td>.5 day</td>
</tr>
<tr>
<td>Product Design</td>
<td>Identify new elements needed by game</td>
<td>designer</td>
<td>2 hours</td>
</tr>
<tr>
<td>Product Design</td>
<td>Identify changes to existing elements</td>
<td>designer</td>
<td>.5 day</td>
</tr>
<tr>
<td>Product Build</td>
<td>Create product-specific implementation of the Game interface</td>
<td>designer</td>
<td>.25 day</td>
</tr>
<tr>
<td>Product Test</td>
<td>Create new .Net projects for libraries and application</td>
<td>developer</td>
<td>.5 hour</td>
</tr>
<tr>
<td>Product Test</td>
<td>Create new make file</td>
<td>Automated/developer</td>
<td>.5 hour</td>
</tr>
<tr>
<td></td>
<td>Modify existing unit tests for existing elements</td>
<td>developer</td>
<td>.5 day</td>
</tr>
<tr>
<td></td>
<td>Execute unit tests</td>
<td>developer</td>
<td>.5 day</td>
</tr>
<tr>
<td></td>
<td>Modify/extend product test suite</td>
<td>tester</td>
<td>1 day</td>
</tr>
<tr>
<td></td>
<td>Execute product tests</td>
<td>tester</td>
<td>.5 day</td>
</tr>
</tbody>
</table>
6.2 Production Resources

The primary resources needed for product production are the Visual Studio .Net environment and the UML modeling tool being used by the organization.

During the analysis and design steps in the product production process the UML model is extended to include any new elements that must be defined.

*The Brickles requirements document includes only those use cases that relate to the Brickles game.*

The .Net environment is used to create any new components that are required. The environment is then used to build an executable.

*A .Net solution has been created for the Brickles game.*

6.3 Bill of materials (BOM)

Here we provide a template BOM. For each product-specific production plan fill in the template with exactly what will be used.

This game requires the following list of code assets:

<table>
<thead>
<tr>
<th>Component</th>
<th>Source</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>product-specific Game component</td>
<td>In-house</td>
<td>30 lines of new code¹</td>
</tr>
<tr>
<td>the generic GameBoard</td>
<td>In-house</td>
<td>40 lines of new code¹</td>
</tr>
<tr>
<td>required Sprites</td>
<td>In-house</td>
<td>20 lines of new code¹</td>
</tr>
<tr>
<td>MovableSprite</td>
<td></td>
<td>15 lines of new code¹</td>
</tr>
<tr>
<td>StationarySprite</td>
<td></td>
<td>0 lines of new code¹</td>
</tr>
<tr>
<td>Puck</td>
<td></td>
<td>5 lines of new code¹</td>
</tr>
<tr>
<td>Paddle</td>
<td></td>
<td>5 lines of new code¹</td>
</tr>
<tr>
<td>Brick</td>
<td></td>
<td>5 lines of new code¹</td>
</tr>
<tr>
<td>Left Wall</td>
<td></td>
<td>5 lines of new code¹</td>
</tr>
<tr>
<td>Right Wall</td>
<td></td>
<td>5 lines of new code¹</td>
</tr>
<tr>
<td>Floor</td>
<td></td>
<td>5 lines of new code¹</td>
</tr>
<tr>
<td>Ceiling</td>
<td></td>
<td>5 lines of new code¹</td>
</tr>
<tr>
<td>BrickPile</td>
<td></td>
<td>25 lines of new code¹</td>
</tr>
<tr>
<td>Implementation of EventHandlerDefinitions interface</td>
<td>In-house</td>
<td>5 lines of new code¹</td>
</tr>
<tr>
<td>BricklesEventHandlerDefinitions</td>
<td></td>
<td>15 lines of new code¹</td>
</tr>
</tbody>
</table>

¹ Since this is an example, only relative costs are given instead of actual costs.
The costs of assets are documented at the product for which they are first used. The profitability of the product is computed differently and amortizes the costs of assets across all of the products that use the asset.

### 6.4 Product-specific details

The rules of the game, which are the most unique parts of the product, are distributed across the Sprites, the EventHandlerDefinitions, and the Game component. Most of the rules enforced by Sprites are common across most, if not all, of the games. The EventHandlerDefinitions component is developed specifically for a game as is the Game component.

The single most important product-specific detail is the animation loop in the Game class. This defines the sequence of events that occur in the Game.

*For Brickles the Game class includes catchers for the exceptions that signal the two different endings to the game.*

### 6.5 Metrics

Two metrics will be important to the product production process: number of new lines of code and the number of unique lines of code.

#### 6.5.1 New lines of code

This metric will describe the number of lines of new code that has to be written for this product. This code may or may not be used in another product later. A higher value for this metric indicates more effort required to produce the product and impacts cost and schedule.

*Since Brickles is the first product in the product line technically all the lines of code are new.*

#### 6.5.2 Unique lines of code

This metric will describe the percentage of a product that is code not used in any other product. This metric will change over time as code assumed to be unique gets reused or as code assumed to be reusable is not reused after some time or product horizon. A higher value of this metric indicates less similarity between products and indicates a longer pay back time.

*The Game module and the event handling definitions, 45 lines of code, are unique to this product.*
7 Attached Processes

7.1 Constructing the Production Plan

[Chastek 02] describes how to build a production plan. We will not repeat that information here. However, in Figure 3 we show a very high-level version of the process in [Chastek 02].

![High-level planning process diagram]

Figure 3 - High-level planning process

\[1\] This section is the “attached process” described in [Clements 02]. For the product line production plan, this process defines how the production plan is built initially. The process focuses mainly on modifying the production plan of the product line.
7.2 Changing the Production Plan

The production plan is reviewed as experience grows and as the asset base is revised. Data is collected at the delivery of each product. The AGM product line organization will review the generic production plan at the end of each increment using the data collected. The review is scheduled after the revision of the core asset base is completed for the next increment. The member of the core asset team that owns the plan initiates the review.

The review is intended to identify any inconsistencies between the newly revised set of assets and the production plan. Prior to the review, the architecture documentation and any information in the plan about the assets should be updated to reflect the new asset base.
Collect data after each product is produced

Increment review/analysis of next products

Modify the plan to reflect changes in core assets

Production plan review

Consistent with core assets?

no

yes

Accept plan

Figure 4 – Process to change the production plan
8 References and Further Reading

For references see the Bibliography document.