(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3714680号
(P3714680)
(45)発行日　平成17年11月9日(2005.11.9)　(24)登録日　平成17年9月2日(2005.9.2)

(51)Int.Cl.[7]　F I
H04N 5/262　H04N 5/262
G06T 3/40　G06T 3/40　C

請求項の数 23　(全 51 頁)

(21)出願番号　特願平8-519371
(86)(22)出願日　平成7年12月6日(1995.12.6)
(65)公表番号　特表平10-511239
(43)公表日　平成10年10月27日(1998.10.27)
(86)国際出願番号　PCT/CA1995/000680
(87)国際公開番号　WO1996/019777
(87)国際公開日　平成8年6月27日(1996.6.27)
審査請求日　平成14年9月13日(2002.9.13)
(31)優先権主張番号　08/361,539
(32)優先日　平成6年12月22日(1994.12.22)
(33)優先権主張国　米国(US)

(73)特許権者
ジェネシス　マイクロチップ　インク
カナダ　オンタリオ　Ｌ４Ｒ　８Ｇ５　マークハム　タウン　センター　ブルヴァード　２００　スウィート　４００
(74)代理人
弁理士　志賀　正武
(74)代理人
弁理士　渡邊　隆
(74)代理人
弁理士　成瀬　重雄
(72)発明者　グレッグゲイン，　ランス
カナダ　オンタリオ　Ｌ４Ｌ　７Ｈ８　ウッドブリッジ　ウーバーン　ドライヴ　３２
最終頁に続く

(54)【発明の名称】デジタル画像ワーピングシステム

(57)【特許請求の範囲】
【請求項１】
ターゲットサンプルの出力シーケンスを生成するために、ソースサンプルの入力シーケンスを受け取ってワーピングするサンプルシーケンスワーピングシステムであって、
ａ）　前記ターゲットサンプルのそれぞれの間の所望のターゲット間隔を表す連続したターゲット増分を受け取り、それに応じて、前記所望のターゲット間隔を有する前記各ターゲットサンプルを発生するために、十分なフィルタ係数を有する各ＦＩＲフィルタを表す連続するフィルタ係数値を発生する第１の手段と、
ｂ）　前記連続するフィルタ係数値を受け取り、それに応じて、前記各ターゲットサンプルを発生するために、前記各ＦＩＲフィルタの所定の左右半分の核を選択する第２の手段と、
ｃ）　前記連続するターゲット増分および前記連続するフィルタ係数値を受け取り、それに応じて、前記ソースサンプルの入力シーケンスから所定数のアップサンプルされた中間サンプルを発生する第３の手段と、
ｄ）　前記ターゲットサンプルの出力シーケンスを発生するために、前記アップサンプルされた中間サンプルを、前記各ＦＩＲフィルタの前記所定の左右半分の核に供給する第４の手段とを具備することを特徴とするサンプルシーケンスワーピングシステム。
【請求項２】
前記第１の手段が、前記各ターゲット増分を、前記各フィルタ係数値および前記ＦＩＲフィルタのサイズと関連づけるためのフィルタ参照テーブルを具備することを特徴とする請

求項１記載のサンプルシーケンスワーピングシステム。
【請求項３】
前記ターゲット増分が０と１の間にあるときに、前記フィルタ参照テーブルが、フィルタ係数値、０を発生することを特徴とする請求項２記載のサンプルシーケンスワーピングシステム。
【請求項４】
前記ターゲット増分が１と２の間にあるときに、前記フィルタ参照テーブルが、フィルタ係数値、１を発生することを特徴とする請求項２記載のサンプルシーケンスワーピングシステム。
【請求項５】
前記ターゲット増分が２と４の間にあるときに、前記フィルタ参照テーブルが、フィルタ係数値、２を発生することを特徴とする請求項２記載のサンプルシーケンスワーピングシステム。
【請求項６】
前記ターゲット増分が４と８の間にあるときに、前記フィルタ参照テーブルが、フィルタ係数値、２を発生することを特徴とする請求項２記載のサンプルシーケンスワーピングシステム。
【請求項７】
前記ターゲット増分が８と１６の間にあるときに、前記フィルタ参照テーブルが、フィルタ係数値、４を発生することを特徴とする請求項２記載のサンプルシーケンスワーピングシステム。
【請求項８】
前記ターゲット増分が１６と３２の間にあるときに、前記フィルタ参照テーブルが、フィルタ係数値、５を発生することを特徴とする請求項２記載のサンプルシーケンスワーピングシステム。
【請求項９】
前記第２の手段が、前記各ターゲットサンプルの両側の前記各ＦＩＲフィルタの最も大きなものの前記所定の左右半分の核を選択する手段をさらに具備することを特徴とする請求項１記載のサンプルシーケンスワーピングシステム。
【請求項１０】
前記第２の手段が、前記各ターゲットサンプルの両側の前記各ＦＩＲフィルタからの前記所定の左右半分の核を選択する手段をさらに具備することを特徴とする請求項１記載のサンプルシーケンスワーピングシステム。
【請求項１１】
前記所定の左右半分の核を選択する手段が、さらに、
ｉ）　前記連続するフィルタ係数値を受け取り、該連続するフィルタ係数値を一サンプル周期だけ遅延させるために、前記第１の手段に接続された第１のレジスタ手段と、
ｉｉ）　前記第１の手段および前記第１のレジスタ手段から出力される２つの連続するフィルタ係数値の最も大きいものを選択し、それに応じて、前記２つの連続するフィルタ係数値の連続する最も大きいものを出力するために、前記第１の手段および前記第１のレジスタ手段に接続された第１の比較器手段とを具備することを特徴とする請求項９記載のサンプルシーケンスワーピングシステム。
【請求項１２】
前記第３の手段が、さらに、
ｉｉｉ）　前記連続するフィルタ係数値を受け取り、該連続するフィルタ係数値を一サンプル周期だけ遅延させるために、前記第１のレジスタ手段に接続された第２のレジスタ手段と、
ｉｖ）　前記第１の手段、前記第１のレジスタ手段および前記第２のレジスタ手段のそれぞれから出力された３つの連続したフィルタ係数値のうちの最も大きいものを選択し、それに応じて、前記３つの連続したフィルタ係数値の連続した最も大きなものを出力するた

めに、前記第１の手段、前記第１のレジスタ手段および前記第２のレジスタ手段に接続された第２の比較器手段と、
ｖ） 前記連続したターゲット増分および前記３つの前記連続したフィルタ係数値の前記連続した最も大きなものを受け取り、それに応じて、連続したアップサンプル増分値を形成するために、前記連続したターゲット増分をそれぞれ、前記３つの前記フィルタ係数値の前記連続した最も大きなものだけ右にシフトするためのアップサンプル増分発生器と、
ｖｉ） 前記連続したバイナリ形式のターゲット増分、前記連続したアップサンプル増分値および前記ソースサンプルの入力シーケンスを受け取り、それに応じて、前記アップサンプル増分値のそれぞれのものだけ間隔をあけて離れた、前記アップサンプル増分値のそれぞれのもので除算した前記ターゲット増分のそれぞれのものに等しい前記所定数のアップサンプルされた中間サンプルを生成するために、前記ソースサンプルの前記入力シーケンスを補間する補間器手段とを具備することを特徴とする請求項１１記載のサンプルシーケンスワーピングシステム。
【請求項１３】
前記第４の手段が、さらに、
ｖｉｉ） 前記２つの連続したフィルタ係数値のうちの前記連続した最も大きなものを受け取って、一のサンプル周期だけ遅延させるための前記第２の比較器手段に接続された第３のレジスタ手段と、
ｖｉｉｉ） 前記２つの連続したフィルタ係数値のうちの前記連続した最も大きなものを受け取って、他のサンプル周期だけ遅延させるための前記第３のレジスタ手段に接続された第４のレジスタ手段と、
ｉｘ） 第１の複数の前記アップサンプルされた中間サンプルを受け取り、該中間サンプルに、一のサンプル周期だけ遅延された前記２つの連続したフィルタ係数値のうちの前記連続した最も大きなものにより同定された前記フィルタ係数の連続したものを乗じ、それに応じて、第１の複数の中間積値を発生し、該第１の複数の中間積値を合計し、それに応じて、一連の累積された中間積値を発生するために、前記第３のレジスタ手段および前記補間器手段に接続された第１の半分のデジタルフィルタ核乗算器アキュムレータ手段と、
ｘ） 第２の複数の前記連続したアップサンプルされた中間サンプルを受け取り、該中間サンプルに、他のサンプル周期だけ遅延された前記２つの連続したフィルタ係数値の前記連続した最も大きなものにより同定された前記フィルタ係数の連続したものを乗じ、それに応じて、第２の複数の中間積値を発生し、これら第２の中間積値を合計し、それに応じて、他の一連の累積された中間積値を発生し、ターゲットサンプルの前記出力シーケンスを発生するために、前記一連の累積された中間積値と前記他の一連の累積された中間積値とを合計すべく、前記第１の半分のデジタルフィルタ核乗算器アキュムレータ手段、前記第４のレジスタ手段および前記補間器手段に接続された第２の半分のデジタルフィルタ核乗算器アキュムレータ手段とを具備することを特徴とする請求項１２記載のサンプルシーケンスワーピングシステム。
【請求項１４】
前記第３の手段が、さらに、
ｉ） 前記連続したターゲット増分と、前記連続したフィルタ係数値とを受け取り、それに応じて、連続したアップサンプル増分値を形成するために、前記連続したターゲット増分を、それぞれ、前記連続したフィルタ係数値だけ右にシフトするためのアップサンプル増分発生器と、
ｉｉ） 前記連続したターゲット増分、前記連続したアップサンプル増分値および前記ソースサンプルの入力シーケンスを受け取り、それに応じて、前記アップサンプル増分値のそれぞれのもので除算した前記ターゲット増分のそれぞれのものに等しい前記所定数の、前記アップサンプル増分値のそれぞれのものだけ間隔をあけたアップサンプルされた中間サンプルを生成するために、前記ソースサンプルの入力シーケンスを補間する補間器手段とを具備することを特徴とする請求項１０記載のサンプルシーケンスワーピングシステム。

【請求項１５】
前記第４の手段が、さらに、
ｉｉｉ）　前記連続したフィルタ係数値を受け取り、これを一のサンプル周期だけ遅延させるために、前記第１の手段に接続されたレジスタ手段と、
ｉｖ）　第１の複数の前記アップサンプルされた中間サンプルを受け取り、これに、一のサンプル周期だけ遅延された前記連続したフィルタ係数値により同定された前記フィルタ係数の連続したものを乗じ、それに応じて第１の複数の中間積値を発生し、これら第１の複数の中間積値を合計し、それに応じて、一連の累積した中間積値を発生するために、前記レジスタ手段および前記補間器手段に接続された第１の半分のデジタルフィルタ核乗算器アキュムレータ手段と、
ｖ）　第２の複数の前記連続したアップサンプルされた中間サンプルを受け取り、これに、前記一のサンプル周期だけ遅延された前記連続したフィルタ係数値により同定された前記フィルタ係数の連続したものを乗じ、それに応じて、第２の複数の中間積値を発生し、これら第２の複数の中間積値を合計し、それに応じて、他の一連の累積した中間積値を発生し、前記ターゲットサンプルの出力シーケンスを発生するために、前記一連の累積した中間積値と前記他の一連の累積した中間積値とを合計すべく、前記第１の半分のデジタルフィルタ核乗算器アキュムレータ手段、前記レジスタ手段および前記補間器手段に接続された第２の半分のデジタルフィルタ核乗算器アキュムレータ手段とを具備することを特徴とする請求項１４記載のサンプルシーケンスワーピングシステム。
【請求項１６】
ターゲットサンプルの出力シーケンスを生成するために、ソースサンプルの入力シーケンスを受け取ってワーピングするサンプルシーケンスワーピング方法であって、
ａ）　前記ターゲットサンプルのそれぞれのものの間の所望のターゲット間隔を表す連続したターゲット増分を受け取り、それに応じて、前記所望のターゲット間隔を有する前記ターゲットサンプルのそれぞれのものを発生するために十分なフィルタ係数を有する各ＦＩＲフィルタを表す連続したフィルタ係数値を発生するステップと、
ｂ）　前記連続したフィルタ係数値を受け取り、それに応じて、前記ターゲットサンプルの前記それぞれのものを発生するために前記各ＦＩＲフィルタの所定の左右半分の核を選択するステップと、
ｃ）　前記連続したターゲット増分および前記連続したフィルタ係数値を受け取り、それに応じて、前記ソースサンプルの入力シーケンスから所定数のアップサンプルされた中間サンプルを発生するステップと、
ｄ）　前記ターゲットサンプルの出力シーケンスを発生するために、前記アップサンプルされた中間サンプルを、前記核ＦＩＲフィルタの前記所定の左右半分の核に適用するステップとを具備することを特徴とするサンプルシーケンスワーピング方法。
【請求項１７】
前記ターゲットサンプルのそれぞれのものを発生するために、前記連続したフィルタ係数値を受け取り、それに応じて、前記各ＦＩＲフィルタの所定の左右半分の核を選択するステップが、さらに、前記各ターゲットサンプルの両側の、前記各ＦＩＲフィルタの最も大きなものの所定の左右半分の核を選択するステップを具備することを特徴とする請求項１６記載のサンプルシーケンスワーピング方法。
【請求項１８】
前記ターゲットサンプルのそれぞれのものを発生するために、前記連続したフィルタ係数値を受け取り、それに応じて、前記各ＦＩＲフィルタの所定の左右半分の核を選択するステップが、さらに、前記各ターゲットサンプルの両側の、前記ＦＩＲフィルタのそれぞれのものから前記所定の左右半分の核を選択するステップを具備することを特徴とする請求項１６記載のサンプルシーケンスワーピング方法。
【請求項１９】
前記所定の左右半分の核を選択するステップが、さらに、
ｉ）　前記連続したフィルタ係数値を受け取り、一のサンプル周期だけ遅延させるステッ

プと、
ｉｉ）　２つの連続したフィルタ係数値の最も大きなものを選択し、それに応じて、前記２つの連続したフィルタ係数値のうちの連続した最も大きなものを出力するステップとを具備することを特徴とする請求項１７記載のサンプルシーケンスワーピング方法。
【請求項２０】
前記連続したターゲット増分と前記連続したフィルタ係数値とを受け取り、それに応じて、前記ソースサンプルの入力シーケンスから所定数のアップサンプルされた中間サンプルを発生するステップが、さらに、
ｉｉｉ）　前記連続したフィルタ係数値を受け取り、他のサンプル周期だけ遅延させるステップと、
ｉｖ）　３つの連続したフィルタ係数値の最も大きなものを選択し、それに応じて、３つの前記連続したフィルタ係数値の連続した最も大きなものを出力するステップと、
ｖ）　前記連続したターゲット増分と、前記３つの前記連続したフィルタ係数値のうちの前記連続した最も大きなものを受け取り、それに応じて、連続したアップサンプルされた増分値を形成するために、前記連続したターゲット増分を前記３つのフィルタ係数値の前記連続した最も大きなものだけ右にシフトするステップと、
ｖｉ）　前記連続したバイナリ形式のターゲット増分、前記連続したアップサンプルされた増分値および前記ソースサンプルの入力シーケンスを受け取り、それに応じて、前記アップサンプルされた増分値のそれぞれのものだけ間隔をあけて離れた、前記アップサンプルされた増分値のそれぞれのものにより除算される前記ターゲット増分のそれぞれのものに等しい前記所定数の、アップサンプルされた中間サンプルを生成するために、前記ソースサンプルの入力シーケンスを補間するステップとを具備することを特徴とする請求項１９記載のサンプルシーケンスワーピング方法。
【請求項２１】
前記ターゲットサンプルの出力シーケンスを発生するために、前記アップサンプルされた中間サンプルを、前記各ＦＩＲフィルタの前記所定の左右半分の核に適用するステップが、さらに、
ｖｉｉ）　２つの前記連続したフィルタ係数値のうちの前記連続した最も大きなものを受け取り、これを一のサンプル周期だけ遅延させるステップと、
ｖｉｉｉ）　２つの前記連続したフィルタ係数値のうちの前記連続した最も大きなものを受け取り、これを他のサンプル周期だけ遅延させるステップと、
ｉｘ）　第一の複数の前記アップサンプルされた中間サンプルを受け取り、これに、一のサンプル周期だけ遅延された２つの前記連続したフィルタ係数値の前記連続した最も大きなものにより同定される前記フィルタ係数の連続したものを乗じ、それに応じて、第１の複数の中間積値を発生し、該第１の複数の中間積値を合計し、それに応じて一連の累積された中間積値を発生するステップと、
ｘ）　第２の複数の前記連続したアップサンプルされた中間サンプルを受け取り、これに、他のサンプル周期だけ遅延された前記２つの連続したフィルタ係数値の前記連続した最も大きなもので同定された前記フィルタ係数の連続したものを乗じ、それに応じて、第２の複数の中間積値を発生し、該第２の中間積値を合計し、これに応じて、他の一連の累積した中間積値を発生し、ターゲットサンプルの前記出力シーケンスを発生するために、前記一連の累積した中間積値と前記他の一連の累積した中間積値を合計するステップとを具備することを特徴とする請求項２０記載のサンプルシーケンスワーピング方法。
【請求項２２】
前記連続したターゲット増分と前記連続したフィルタ係数値とを受け取り、それに応じて、前記ソースサンプルの入力シーケンスから、所定数のアップサンプルされた中間サンプルを発生するステップが、さらに、
ｉ）　前記連続したターゲット増分と前記連続したフィルタ係数値とを受け取り、それに応じて、連続したアップサンプル増分値を形成するために、前記連続したターゲット増分をそれぞれ前記連続したフィルタ係数値だけ右にシフトするステップと、

ｉｉ） 前記連続したターゲット増分、前記連続したアップサンプル増分値および前記ソースサンプルの入力シーケンスを受け取り、それに応じて、前記アップサンプルされた増分値のそれぞれのものだけ間隔をあけて離れた、前記アップサンプル増分値のそれぞれのもので除算された前記ターゲット増分のそれぞれのものに等しい所定数の、アップサンプルされた中間サンプルを生成するために、前記ソースサンプルの入力シーケンスを補間するステップとを具備することを特徴とする請求項１８記載のサンプルシーケンスワーピング方法。

【請求項２３】

前記ターゲットサンプルの出力シーケンスを発生するために、前記アップサンプルされた中間サンプルを、前記各ＦＩＲフィルタの前記所定の左右半分の核に適用するステップが、さらに、

ｉｉｉ） 前記連続したフィルタ係数値を受け取り、これを一のサンプル周期だけ遅延させるステップと、

ｉｖ） 第１の複数の前記アップサンプルされた中間サンプルを受け取り、これに、一のサンプル周期だけ遅延された前記連続したフィルタ係数値により同定される前記フィルタ係数の連続したもの乗じ、これに応じて、第１の複数の中間積値を発生し、該第１の複数の中間積値を合計し、それに応じて、一連の累積した中間積値を発生するステップと、

ｖ） 第２の複数の前記連続したアップサンプルされた中間サンプルを受け取り、これに、前記一のサンプル周期だけ遅延された前記連続したフィルタ係数値により同定される前記フィルタ係数の連続したものを乗じ、これに応じて、第２の複数の中間積値を発生し、該第２の複数の中間積値を合計し、それに応じて、他の一連の累積した中間積値を発生し、前記ターゲットサンプルの出力シーケンスを発生するために、前記一連の累積した中間積値と前記他の一連の累積した中間積値とを合計するステップとを具備することを特徴とする請求項２２記載のサンプルシーケンスワーピング方法。

【発明の詳細な説明】

発明の分野

この発明は、概括的に、デジタル信号処理に関し、さらに詳細には、ピクセルおよび／またはラインおよび／またはフレームの入力ストリームから、ワーピングされたピクセルおよび／またはラインおよび／またはフレームの出力ストリームをそれぞれ発生するための画像ワーピングシステムに関するものである。

発明の背景

デジタル画像ワーピングは、規則正しく間隔をあけたソース画像を、該ソース画像とは異なる間隔を有するターゲット画像を生成するために、動的にリサンプリングする信号処理である。多くの現代の映画は、例えば、遠近法、奇妙な形状をした画像境界およびテクスチャマッピングのような特殊効果を生成するためにデジタルワーピングを使用している。従来技術のリアルタイム画像ワーピングシステムは、リアルタイムで膨大な数の計算を実施しなければならないために、非常に複雑かつ高価である。従来のシステムは、ワーピングされた画像の空間周波数をナイキストリミットまで動的に帯域制限(bandlimit)せず、望ましくないエーリアシング歪みを生じていた。公知の従来のデジタル画像ワーピングシステムの例は、以下の刊行物および特許において説明されている。

I. Wolberg, George,"Separable Image Warping: Implications and Techniques", Ph. D. Thesis, Dept. of Computer Science, Columbia University, NY, 1990, および"Digital Image Warping", IEEE Computer Society Press, Los Alamitos, CA, 1990;

II. 米国特許第５３５５３２８号明細書(アルベイタ(Arbeitar)他)

発明の概要

この発明によれば、固有のナイキスト帯域制限で、水平方向および垂直方向の両方に、リアルタイムで動的な拡大および縮小を実施し、かつ、外部ライン記憶装置または他のメモリを必要としない画像ワーピングシステムが提供される。また、この発明は、レジスタの代わりに外部フレーム記憶装置を用いることにより、複数画像を時間的にワーピング（例えば、速い動作と遅い動作との間の動的な変更）することができる。したがって、この発

明のシステムは、公知の従来技術に対してかなり進歩したものである。この発明により、ワーピングした画像を生成するために、３つの異なる処理が連続して実施される。第１の処理は、所望のターゲットピクセル（またはラインまたはフレーム）間隔を、適当なＦＩＲフィルタに写像する。ここで、ＦＩＲフィルタの大きさは、所望のターゲットピクセル（またはラインまたはフレーム）間隔の大きさに基づいて選択される。フィルタが一旦選択されると、第２の処理が、入力ソースピクセル（またはラインまたはフレーム）に基づいて、適当な数のアップサンプルされたピクセル（またはラインまたはフレーム）を生成する。最後の処理は、第１の処理による正確なフィルタ係数と、第２の処理による補間されたピクセル（またはラインまたはフレーム）とを乗じ、その結果、正確にナイキスト帯域制限されたターゲットピクセル（またはラインまたはフレーム）を得る。

このように、この発明の一側面によれば、出力ピクセルおよび／またはラインおよび／またはフレーム間の所望の間隔を定める、ユーザにより定義されるターゲット増分にしたがって、ワーピングされたピクセルおよび／またはラインおよび／またはフレームの出力ストリームを生成するために、ピクセルおよび／またはラインおよび／またはフレームのいずれか１つの入力ストリームをデジタル式にワーピングする装置および方法が提供される。デジタル式にワーピングされたピクセルおよび／またはラインの場合には、得られる画像は、空間的にワーピングされたものと考えられる。デジタル式にワーピングされたフレームの場合には、得られる一組のフレームは時間的にワーピングされたものと考えられる。

この発明の第１の実施形態においては、システムは、ターゲット増分を特定のフィルタサイズに写像するためのフィルタ参照テーブルと、一対の比較器への入力として前記フィルタ参照テーブルから出力された３つの連続したフィルタサイズを補給する一対のレジスタとを具備している。前記比較器のうちの第１のものは、フィルタ参照テーブルおよび両レジスタにより出力される３つの値から最も大きなフィルタサイズを決定するものである一方、第２の比較器は、フィルタ参照テーブルおよび第１のレジスタから出力される最も大きなフィルタサイズを決定するものである。第１の比較器の出力は、フィルタサイズをフィルタ係数に写像し、その後、アップサンプル増分を発生するために、ターゲット増分をこの量だけ右にシフトするアップサンプル増分発生器により受け取られる。このアップサンプル増分は、それに応じて、平均的に間隔をあけた適当な数のアップサンプルピクセル値を発生する補間器による受け取られる。第２の比較器からの出力は、左右半分の核フィルタ発生器にそれぞれ接続されたさらに２つのレジスタを有するパイプライン内に格納される。この左右半分の核フィルタ発生器は、補間器からのアップサンプルされたピクセルを受け取り、ワーピングされた出力ピクセルを生成するために、受け取ったピクセルにそれぞれの半分の核係数を乗じる。

この発明の第２の実施形態においては、システムは、ターゲット増分を特定のフィルタサイズに写像するためのフィルタ参照テーブルと、左右半分の核フィルタ発生器への入力として、フィルタ参照テーブルからのフィルタサイズを保持する単一のレジスタとを具備している。フィルタ参照テーブルの出力は、フィルタサイズをフィルタ係数に写像し、その後、アップサンプル増分を発生するために、ターゲット増分を、この量だけ右にシフトするアップサンプル増分発生器により受け取られる。アップサンプル増分は、それに応じて、平均的に間隔をあけた適当な数のアップサンプルピクセル値を発生する補間器により受け取られる。左右半分のフィルタ核発生器は、補間器からのアップサンプルされたピクセルを受け取って、ワーピングされた出力ピクセルを生成するために、受け取ったピクセルにそれぞれの半分の核係数を乗じる。

各実施形態は、レジスタをライン記憶装置およびフレーム記憶装置にそれぞれ置き換えることにより、ラインおよび／またはフレームをデジタル式にワーピングするように拡張することができる。これらの大きな記憶装置ユニットは、以下に詳しく説明される実施形態におけるレジスタの機能と同等の機能を奏する。

この発明のワーピング技術は、全てがジェネシスマイクロチップ社(Genesis Microchip Inc.)に譲渡され、その内容が参照によりこの明細書に組み込まれている出願係属中の米国

特許出願第０８／１２６３８８号、第０８／１２５５３０号、第０８／１７２０６５号、第０８／１２４２０１号に説明されている画像フィルタリングおよび画像拡張装置および方法を用いることにより実施できる。

さらに、この発明に係るデジタル画像ワーピングシステムは、ソース画像を、任意の、きまったターゲット画像サイズにリサイジングする方法に関する、上述した米国特許第５３５５３２８号明細書に記載された技術に基づいている。この従来技術のシステムによれば、各ターゲットピクセルおよび／またはラインおよび／またはフレームは、ソースピクセルおよび／またはラインおよび／またはフレームに対して同じ量でリサイズされる。この発明によれば、各ターゲットピクセルおよび／またはラインおよび／またはフレームは、ソースピクセルおよび／またはラインおよび／またはフレームに対して任意に、個別にサイジングされる。さらに詳細には、この発明によれば、ソースピクセルおよび／またはラインおよび／またはフレームは、ターゲットピクセルおよび／またはラインおよび／またはフレームのリサイズファクタの変更に基づいて、動的にアップサンプルされる。ソースピクセルおよび／またはラインおよび／またはフレームは常に最も近い１オクターブにアップサンプルされ、その後、ナイキスト帯域制限されたターゲットピクセルおよび／またはラインおよび／またはフレームを生成するためにきまった大きさのＦＩＲフィルタによりデシメート(decimate)される。フィルタサイズは、各ターゲットピクセルおよび／またはラインおよび／またはフレームごとに、ユーザにより指定されたような空間的な位置決めまたは時間的な位置決めに基づいて動的に決定される。この発明によれば、きまった大きさのＦＩＲフィルタを使用することによって、従来のビデオワーピングシステムと比較して大幅なシリコンコストの削減を達成することができる。

【図面の簡単な説明】

好ましい実施形態は、以下の図面を参照しながら以下に詳細に説明される。

図１は、複数のソースピクセルから提供される複数のターゲットピクセルを示しており、ここでは、ターゲットピクセルは、ユーザの決定したターゲット増分にしたがって間隔をあけている。

図２は、この発明に係るデジタル画像ワーピングシステムの機能的な構成要素を示すブロックダイアグラムである。

図３は、ユーザの決定したターゲット増分を動的に変更するためのソースおよびターゲットピクセルの相対的な空間的位置決めと、代表的な例にしたがって、図２のシステムを用いてターゲットピクセルを発生するための適当なＦＩＲフィルタの選択とを示している。

図４は、この発明の好適な実施形態に係るデジタル画像ワーピングシステムの機能的な構成要素を示すブロックダイアグラムである。

図５は、ユーザの決定したターゲット増分を動的に変更するためのソースおよびターゲットピクセルの相対的な空間的位置決めと、図４の好適な実施形態を用いてターゲットピクセルを発生するための適当なＦＩＲフィルタの選択とを示している。

好ましい実施形態の詳細な説明

以下の説明において、好ましい実施形態および他の実施形態は、ピクセルおよび／またはラインのワーピングに等しく適用される。同様の発明の原理は、図２および図４において、各レジスタをフレーム記憶装置に置き換えることにより、フレームを時間的にワーピングする場合に用いることができる。さらに、この発明はビデオ画像のワーピングに限定されるものではない。例えば、１次元ワーピングの場合には、この発明の原理は、ビデオサンプルよりむしろ音に（例えば、デジタルオーディオテープドライブのジターを補正するために）適用され得る。

上述したように、画像のデジタルワーピングは、ソースピクセル間隔に対するターゲットピクセル間隔により明記される。各ターゲットピクセル間の水平方向の間隔および各ターゲットライン間の垂直方向の間隔は、ユーザにより指定されなければならない。同様に、画像の連続するフレーム間の時間的な間隔（すなわち時間）は、ユーザにより指定されなければならない。この詳細な情報は、非常に複雑なワーピングを生成するために使用され得る非常に詳細な制御レベルを提供する。ユーザが決定するターゲット間隔の発生は、当

業者に公知であり、この発明の部分を構成するものではない。この発明のシステムは、ソースピクセル（および／またはラインおよび／またはフレーム）のストリームおよびユーザの指定した間隔を決定する所望のターゲット増分（ＴａｒＩｎｃ）を入力として受け取る。

図１は、ユーザの定めたターゲット増分（ＴａｒＩｎｃ）に基づくソースピクセルからのターゲットピクセルの発生を示している。簡単のために、この開示の残りの部分は、ソースピクセルからのターゲットピクセルの発生に言及する。全く同一の方法論および回路が、ピクセルをビデオラインに代えることにより垂直な処理に、そして、ピクセルをビデオフレームに代えることにより時間的な処理に適用できることは理解されるであろう。

図１に見られるように、ワーピングされた画像の最初および最後のターゲットピクセルは、最初および最後のソースピクセルと空間的に整列されている。しかしながら、中間のターゲットピクセルの間隔は、ターゲット増分（ＴａｒＩｎｃ）によって決定されている。特に、図１に示された例では、最初の（すなわち、一番左の）中間ターゲットピクセルは、最初のターゲットピクセルから、ソースピクセル間の間隔の１．５倍に等しい間隔をあけており、２番目の中間ターゲットピクセルは、第１の中間ターゲットピクセルから、ソースピクセル間の間隔の０．７５倍に等しい間隔をあけている。一方、最後のターゲットピクセルは、第２の中間ターゲットピクセルから、ソースピクセル間の間隔の０．７５倍に等しい間隔をあけて離れており、総間隔は３．０（すなわち、最初と最後のソースピクセル間の間隔）である。

ここで、図２を見ると、動的に変化するターゲット増分にしたがって、ソースピクセルのストリームからワーピングされた出力ピクセルのストリームを発生させるための、この発明の第１の実施形態に係る機能的なブロックダイアグラムが示されている。

第１に、ターゲット増分（ＴａｒＩｎｃ）は、以下の表Ａに示されたマッピングに基づいたフィルタ係数値により表されるような特定のＦＩＲフィルタサイズに、フィルタ参照テーブル２１を介して写像される。ＦＩＲ係数は、読み出し専用テーブル（すなわちＲＯＭ）に格納されるか、または、ユーザにより提供され得る。係数のプログラミングにより、ＦＩＲフィルタを調整する柔軟性がユーザに与えられる。フィルタサイズは、ＴａｒＩｎｃに基づいてフィルタ参照テーブルをインデクシングすることにより選択される。表Ａが単に１つの実施例を示しているだけであることに注意することは重要である。この表は、全てのサイズのＴａｒＩｎｃをサポートしかつ全てのフィルタサイズおよびフィルタ係数を発生するように拡張され得る。

表Ａ　ＦＩＲフィルタテーブルおよびフィルタ係数

TarInc(t)	0<t<1	1<t<2	2<t<4	4<t<8	8<t<16	16<t<32
ﾌｨﾙﾀｻｲｽﾞ	3ﾀｯﾌﾟ	5ﾀｯﾌﾟ	9ﾀｯﾌﾟ	17ﾀｯﾌﾟ	33ﾀｯﾌﾟ	65ﾀｯﾌﾟ
ﾌｨﾙﾀ係数	0	1	2	3	4	5

ＴａｒＩｎｃが、一旦写像されると、ターゲットピクセルの両側の最も大きなフィルタが、そのピクセルのフィルタとして選択される。図３はこのステップを示している。図３の第５の線（ＴａｒＩｎｃ→フィルタマッピング）は、上記表Ａの写像に基づいて、第４の線（ターゲット増分）からのＴａｒＩｎｃ値を種々のＦＩＲフィルタに写像したものを示している。このことは、ターゲットピクセルの両側に表されているフィルタサイズのうちの最も大きなフィルタサイズ（すなわちタップ数）をターゲットピクセル内に配置し、そ

れがそのピクセルのために選択されたフィルタであることを示すことにより、図３に概略的に表されている。図３において、第３の線（ターゲット）は、９タップフィルタにより発生される第１のターゲットピクセル、９タップフィルタにより発生される第２のターゲットピクセル、３タップフィルタにより発生される第３のターゲットピクセル等を示している。

上述したように、ＴａｒＩｎｃは、マッピング表Ａに従うＦＩＲフィルタサイズに、フィルタ参照テーブル２１を介して写像され、その出力（すなわち、フィルタ係数値）は、それぞれ連続してＡおよびＢレジスタ２２，２３に、連続するクロックサイクルにおいて格納され、それによってパイプラインを形成する。比較器２４は、ターゲットピクセルの両側の隣接する２つのフィルタのうちのいずれがより大きいかを決定し、このフィルタの表示（すなわち、関連するフィルタ係数）を比較器２４の出力側に接続された左レジスタ２５に格納する。左核のＭＡＣ２６（すなわち、乗算器アキュムレータ(Multiplier ACcumulator)）は、左レジスタ２５内に格納されているフィルタ係数値を読み込み、指定されたフィルタ核の左半分を用いてターゲットピクセル値の半分を発生する。次のクロックサイクルにおいては、右核のＭＡＣ２７が、右レジスタ２８に格納されているフィルタ核の右半分および中央タップを用いてターゲットピクセル値の残りを発生し、その結果の値を、左核のＭＡＣ２６から受け取った出力値と合計し、ワーピングされた出力ピクセルを得る。

比較器２４、左右レジスタ２５，２８、および左右核のＭＡＣ２６，２７に関連して上述したように、動的フィルタリングが実施される前に、アップサンプルされたピクセルの正確な数が発生されなければならない。再度、図２を参照すると、アップサンプルされたピクセルの数は、ＡおよびＢレジスタ２２，２３の前後に格納された最も大きなフィルタサイズにより決定される。

比較器２９は、最も大きなフィルタサイズ（すなわち、フィルタ参照テーブル２１からの最も大きなフィルタ係数出力）を決定し、その値をアップサンプル増分発生器３０に転送する。アップサンプル増分発生器は、ＴａｒＩｎｃ値を受け取って、アップサンプル増分値を生成するために、受け取ったＴａｒＩｎｃ値を受け取ったフィルタ係数に等しい所定のビット数だけ右にシフトする。図３に示す例では、第１のターゲット増分は２．２５であり、“４で割る”操作（すなわち、ＴａｒＩｎｃを２ビット右にシフトすることにより、アップサンプル増分値０．５６２５を得る）に対応するフィルタ係数２を得る。そのアップサンプル増分値（すなわち、右にシフトしたＴａｒＩｎｃ値）は、要求されたサイズのＦＩＲフィルタ（例えば、ＴａｒＩｎｃ＝２．２５に対して９タップフィルタ）への適用のためのアップサンプルされたピクセルの正確な等しい間隔を表している。

補間器３１は、アップサンプル増分値を読み込み、ソースピクセルからアップサンプルされたピクセルを、連続したベース上に、アップサンプル増分値により決定される間隔で、内部インクリメンタまたはカウンタ値がＴａｒＩｎｃに達するまで補間する。上述されかつ図３に示された例においては、９タップＦＩＲフィルタへの適用のために、ソースピクセル間隔に対して０．５６２５の間隔で、４つのアップサンプルされたピクセルが発生される。補間器３１は、周知の線形補間器、本出願人の係属中の特許出願第０８／１７２０６５号に開示されたような二次補間器、または任意の他の好適な設計の補間器として装備されてもよい。

補間器３１により発生されるアップサンプルされたピクセルは、正確な係数が正確なアップサンプルされたピクセルに乗算されることを保証する制御ロジックを有する左右核のＭＡＣ２６，２７に転送される。このように、図３において、線２（アップサンプル）は、ターゲットピクセルを発生するように装備された異なるサイズのＦＩＲフィルタに対して要求される可能な限り多くの数のアップサンプルピクセルを表している。図３の線２と線３との間に示されたＦＩＲフィルタの動作は、正確な係数を正確なアップサンプルされたピクセルに乗じる左右核のＭＡＣ２６，２７により実施される。左右核のＭＡＣ２６，２７の内部制御ロジックは、アップサンプルされたピクセルの所定のものが、係数とアップサンプルされたピクセルとを正確に整合するためにスキップされることを保証している。

例えば、第１の中間ターゲットピクセルを発生するために用いられる９タップＦＩＲフィルタが、線２からアップサンプルされた全てのピクセルを用いるのに対して、隣接する３タップフィルタは、第５，９，第１１の発生されたアップサンプルされたピクセルのみを用い、第６，第７，第８，第１０のアップサンプルされたピクセル、その他をスキップしている。

左右核のＭＡＣ２６，２７の制御ロジックは、本件出願人の出願係属中の米国特許出願第０８／１２６３８８号に説明されているように、システムの始動時に、第１のターゲットピクセルについての右半分の核値を２倍にすることにより、第１のターゲットピクセルをも発生する。最後のターゲットピクセルは、左核値を２倍することにより発生される。この倍加は、全てのＦＩＲ係数が単一のゲインを維持することを保証するために要求される。

ここで、図４を見ると、ターゲット増分の動的な変化に従って、ソースピクセルのストリームから、ワーピングされた出力ピクセルのストリームを発生させるための好ましい実施形態に係る機能的なブロックダイアグラムが示されている。

最初に、ターゲット増分（ＴａｒＩｎｃ）は、フィルタ参照テーブル４１を介して、特定のＦＩＲフィルタサイズに、上記表Ａに示されたマッピングに基づくフィルタ係数値により表されるように写像される。ＦＩＲ係数は、読み出し専用テーブル（すなわちＲＯＭ）に格納され、または、ユーザにより提供され得る。フィルタサイズは、ＴａｒＩｎｃに基づいて係数テーブルをインデクシングすることにより選択される。表Ａが、単に１つの実施例を示しているに過ぎないことに注意することは重要である。この表は、全てのサイズのＴａｒＩｎｃをサポートし、全フィルタサイズおよびフィルタ係数を発生するように拡張され得る。

ＴａｒＩｎｃが一旦写像されると、フィルタサイズはレジスタ４２内に保持されて、左右核の乗算器アキュムレータ４４，４５に供給される。図５は、このステップを示している。図５の第５の線（ＴａｒＩｎｃ→フィルタマッピング）は、上記表Ａのマッピングに基づいて第４の線（ターゲット増分）からのＴａｒＩｎｃ値を種々のＦＩＲフィルタに写像したものを示している。これは、非対称フィルタの各半分の核のサイズをターゲットピクセルの各側に配置し、該ピクセルに対して各半分の核を示すことにより、図５に概略的に表されている。図５において、第３の線（ターゲット）は、９タップフィルタにより発生される第１のターゲットピクセル、９タップフィルタの半分および３タップフィルタの半分からなる非対称フィルタにより発生される第２のターゲットピクセル、３タップフィルタにより発生される第３のターゲットピクセルおよび３タップフィルタの半分および５タップフィルタの半分からなる非対称フィルタにより発生される第４のターゲットピクセルを示している。

上述したように、ＴａｒＩｎｃは、フィルタ参照テーブル４１を介して、マッピング表Ａに従ってＦＩＲフィルタサイズに写像され、その出力（すなわち、フィルタ係数値）は、連続してレジスタ４２内に格納される。左核のＭＡＣ４４（すなわち、乗算器アキュムレータ）は、レジスタ４２内に格納されたフィルタ係数値を読み取り、半分のターゲットピクセル値と、指定されたフィルタ核の左半分および左側半分の核のためのセンタータップを用いたセンタータップ積値への第１のコントリビューション(contribution)とを発生する。同じクロックサイクルにおいて、右核のＭＡＣ４５は、他の半分のターゲットピクセル値と、右半分のフィルタ核および右側半分の核のためのセンタータップを用いたセンタータップ積値への第２のコントリビューションとを発生する。右核のＭＡＣ４５は、その後、センタータップ乗算積への第１および第２のコントリビューションを合計し、その合計した値を、平均センタータップ乗算積を得るために２で割り、最後にその平均センタータップ乗算積と、左核のＭＡＣ４４から受け取ったターゲットピクセル値の左半分およびターゲットピクセル値の右半分とを合計し、最終的なワーピングされた出力ピクセルを得る。

センタータップ乗算積への第１および第２のコントリビューションを合計し、その合計値を２で割ることに代えて、各半分の核ごとのセンタータップ係数は、単に、左右核のＭＡ

Ｃ４４，４５の関連する係数ＲＯＭに格納される前に、または、この発明がソフトウェア内に装備された適当なコードを介して、二等分されてもよい。
アップサンプル増分発生器４６は、ＴａｒＩｎｃ値を受け取り、かつ、アップサンプル増分値を生成するために、受け取ったフィルタ係数に等しい所定のビット数だけ右にシフトする。図５の例では、第１のターゲット増分は２．２５であり、“４で割る”操作（すなわち、ＴａｒＩｎｃを右に２ビットシフトすることによりアップサンプル増分０．５６２５が得られる。）に相当するフィルタ係数、２が得られる。アップサンプル増分値（すなわち、右にシフトされたＴａｒＩｎｃ値）は、要求された大きさのＦＩＲフィルタ（例えば、ＴａｒＩｎｃ＝２．２５に対して９タップフィルタ）への適用のためのアップサンプルされたピクセルの正確な等しい間隔を表している。
補間器４７は、アップサンプル増分値を読み込み、ソースピクセルからのアップサンプルピクセルを連続するベース上に、アップサンプル増分値により決定された間隔で、内部インクリメンタまたはカウンタ値がＴａｒＩｎｃに達するまで補間する。上述されかつ図５に示された例では、９タップＦＩＲフィルタへの適用のために、ソースピクセル間隔に対して０．５６２５の間隔を有する４つのアップサンプルピクセルが発生される。補間器４７は、周知の線形補間器、本件出願人の係属中の米国特許出願第０８／１７２０６５号に開示されたような二次補間器、または、任意の他の好適に設計された補間器として装備されてもよい。
補間器４７により発生されたアップサンプルされたピクセルは、正確なアップサンプルされたピクセルに正確な係数が乗じられることを保証するための制御ロジックを有する左右核のＭＡＣ４４，４５に転送される。図５の線２と線３との間におけるＦＩＲフィルタの図示された動作は、正確な係数を正確なアップサンプルされたピクセル値に乗じる左右半分の核のＭＡＣ４４，４５により実行される。
左右核のＭＡＣ４４，４５の制御ロジックは、出願第０８／１２６３８８号に開示された方法で、最初のターゲットピクセルについての右半分の核値を２倍することによりシステムの始動時の第１のターゲットピクセルをも発生する。最後のターゲットピクセルは、左核値を２倍することにより発生される。この倍加は、全てのＦＩＲフィルタが単一のゲインを維持することを保証するために要求される。
この発明の他の実施形態および変更は可能である。例えば、さらに他の実施形態により、この発明は、Ｃ言語を用いたソフトウェアで実施される。この他のソフトウェア実施形態は、この明細書の付録Ａに十分に開示されている。好ましい実施形態にしたがって実施されるワーピングは、固有ナイキスト帯域制限を提供することとして説明されているけれども、ナイキスト帯域制限なしに（例えば、ソフトフォーカスワーピング）、この発明の原理を用いて特殊効果ワーピングが実施されてもよい。言い換えると、プレフィルタリングおよび補間ステップは、ナイキスト帯域制限なしに実施され得る。全てのそのような他の実施形態および変更は、添付された請求の範囲に記載された発明の範囲内に配されるものである。

付録 A

```
1://
2:// Vertical and Horizontal Genesis Warping Software 1.0
3://
4:// This program warps an input Targa file in both dimensions based on two
5:// accompanying vertical and horizontal Warp Containment Files.
6://
7:// Program Name: WARP.CPP
8://
9:// Author: James D. E. Goel
10://
11://      - Initial release of the software.
12://   If you have any problems, please contact me.
13://
14:// Copyright (c)  1993 Genesis Microchip Inc.
15://                                        200 Town Centre Blvd., Suite 400
16://                                        Markham, Ontario
17://                                        L3R 8G5
18://                                        (905) 470-2742
19://
20://      All rights reserved.
21://
22:// Modifications:
23://
24:// Aug. 18, 1994  - James Goel
25://                                        - Initial release
26://
27:// Setup standard C definitions
28://
29:#include <stdio.h>
30:#include <stdlib.h>
31:#include <math.h>
32:#include <string.h>
33:#include <assert.h>
34:#include <afx.h>
35:#include "image.h"
36:#include "logic.h"
37:
38://
39:// Create a number of useful type definitions
40://
41:typedef unsigned char BYTE;
42:typedef unsigned char PIXEL;
43:typedef unsigned long ULONG;
44:typedef unsigned int UINT;
45:
46://
47:// This header structure is used to define a
48:// simple binary format for reading and storing
49:// image data.
50://
51:typedef struct image_hdr
52:{
53:      //
54:      // Image Width and Height
55:      //
56:      unsigned int Width;
57:      unsigned int Height;
58:      //
59:      // Image type
60:      // There are various different Genesis define
61:      // images types:
62:      // 0 = Red Pixels
63:      // 1 = Green Pixels
64:      // 2 = Blue Pixels
65:      // 3 = Luma Pixels
66:      // 4 = Chroma (U/V time multiplexed)
67:      // 5 = Black & White Pixels
68:      //
```

```
69:     int image_type;
70:} Genesis_Header;
71:
72://
73:// This header structure is used to provide a
74:// simple and consistent method for parsing
75:// command line parameters.
76://
77:typedef struct cmd_line_tag
78:{
79:     char     pchVTarInc[13];// Vertical warp containment field (VTarInc)
80:     char     pchHTarInc[13];// Horizontal warp containment field (HTarInc)
81:     char     pchSrc[13];                    // Image to be warped
82:     char     pchWarp[13];          // Output warped image
83:} tCmd_Line;
84:
85://
86:// ************************************************
87:// usage -- Prints appropriate command line usage
88:// ************************************************
89://
90:void Usage()
91:{
92:     //
93:     // Print out the proper usage
94:     //
95:     printf ( "ERROR: Incorrect command line parameters.\n");
96:     printf ( "Usage: WARP -v [Vertical Warp Containment File] ");
97:     printf ( "-h [Horizontal Warp Containment File]\n");
98:     printf ( "          -w [Image to be Warped] ");
99:     printf ( "-o [Output Warped Filename]\n");
100:    printf ( "Please do not use any wild card characters.\n\n");
101:}
102://
103:// ************************************************
104:// Parse_Cmd -- Parses the command line parameters
105:// ************************************************
106://
107:void Parse_Cmd( int argc, char *argv[], tCmd_Line *psCmd_Line)
108:{
109:     CFileException* theException = new CFileException;
110:     int          CurrArg=0;                              // Command line argument counter
111:     //
112:     // IF there are no arguments supplied to the main line program, THEN
113:     //
114:     if (argc == 1)
115:     {
116:          //
117:          // Print usage information
118:          //
119:          Usage();
120:          THROW( theException );
121:     //
122:     // END IF -- file arguments
123:     //
124:     }
125:     //
126:     // WHILE reading through the command line arguments,
127:     //
128:     while ( ++CurrArg < argc )
129:     {
130:          //
131:          // IF the input switches do not use dashes (-), THEN
132:          //
133:          if ( argv[CurrArg][0] != '-' )
134:          {
135:               //
136:               // Pring usage information
137:               //
138:               Usage();
139:               THROW( theException );
```

```
//
// END IF – input switches
//
}
//
// SWITCH ON: command line switches
//
switch ( argv[CurrArg][1] )
{
        //
        // ON CASE: Vertical Warp Containment filename
        //
        case 'V':
        case 'v':
                //
                // Copy the warp containment filename
                //
                strcpy( psCmd_Line->pchVTarInc, argv[++CurrArg] );
        //
        // END CASE – Copy Straight Grid Filename
        //
        break;
        //
        // ON CASE: Horizontal Warp Containment filename
        //
        case 'H':
        case 'h':
                //
                // Copy the warp containment filename
                //
                strcpy( psCmd_Line->pchHTarInc, argv[++CurrArg] );
        //
        // END CASE – Copy Straight Grid Filename
        //
        break;
        //
        // ON CASE: Image to be warped
        //
        case 'W':
        case 'w':
                //
                // Copy source image filename
                //
                strcpy( psCmd_Line->pchSrc, argv[++CurrArg] );
        //
        // END CASE – Copy source image filename
        //
        break;
        //
        // ON CASE: Output warped image
        //
        case 'O':
        case 'o':
                //
                // Copy output warped image filename
                //
                strcpy( psCmd_Line->pchWarp, argv[++CurrArg] );
        //
        // END CASE – Output warped image
        //
        break;
        //
        // ON CASE DEFAULT: Show proper command line usage
        //
        default:
                //
                // Print proper usage
                //
                Usage();
                THROW( theException );
        //
```

```
211:                        // END CASE – default
212:                        //
213:                        break;
214:                //
215:                // END SWITCH
216:                //
217:                }
218:    //
219:    // END WHILE
220:    //
221:    }
222://
223:// End Parse Command Line
224://
225:}
226://
227:// ************************************************************************
228:// Main program loop
229:// ************************************************************************
230://
231:void main(int argc, char *argv[])
232:{
233:    // Exception object for debugging
234:    CFileException* theException = new CFileException;
235:    Image                   SrcImage;           // Source image to be warped
236:    Image                   TarImage;           // Output warped image
237:    Pixel                   VUpPixel;           // Vertical Up sampled pixel
238:    Pixel                   VFiltPix;           // Vertical Filtered pixel
239:    LeftKernel      VLeftKernel;        // Vertical Left FIR Filter Kernel MAC
240:    RightKernel     VRightKernel;       // Vertical Right FIR Filter Kernel MAC
241:    FILE                    *pfVTarInc;         // File pointer to the Vertical TarInc
242:    UINT                    iVTarIncWidth;      // Width of the the containment file
243:    UINT                    iVTarIncHeight;     // Height of the containment file
244:    UINT                    iTarType;           // *** Reserved for future expansion ***
245:    float                   *pVTarIncLine;      // Pointer to a vertical warp TarInc
246:    float                   *pVTarIncLineTmp;   // Copy of the previous pointer
247:    tCmd_Line       sCmd_Line;          // Holds the command line parameters
248:    LOOKUP          VFiltLookup;        // Maps the Vert TarInc to FIR filter size
249:    IncGen          VUpIncGen;          // Object to generate a Vertical Up Sample increment size
250:    Interpolate     VInterpolator;      // Object to generate an interpolated vertical up-sampled pixel
251:    BYTE                    *pSrcCol;           // Pointer to a source image column
252:    BYTE                    *pTarCol;           // Pointer to a target image column
253:    BYTE                    *pTarColTmp;        // Copy of the previous pointer
254:    UINT                    iColumn;            // Counts the number of columns in the input
image
255:    UINT                    iCount;             // Counts the number of output pixels generated
256:    //
257:  // Print introduction
258:    //
259:    fprintf ( stderr, "\n\nWARP FAST!  Generates a warped Targa file.  Aug-23-1994\n");
260:    fprintf ( stderr, "Copyright(c) 1994 Genesis Microchip. All Rights Reserved\n");
261:    fprintf ( stderr, "Author: James Goel\n\n");
262:    //
263:    // Parse the command line parameters
264:    // Load the command line parameters into the command line
265:    // structure
266:    //
267:    TRY
268:    {
269:            Parse_Cmd( argc, argv, &sCmd_Line );
270:    }
271:    CATCH( CException, theException )
272:    {
273:            TRACE( "Exception: Error during command parsing phase...\n" );
274:            exit(1);
275:    }
276:    END_CATCH
277:    //
278:    // ************************************************************************
279:    // VERTICAL WARPING ALGORITHM
280:    // ************************************************************************
```

```
281:    //
282:    // Open the Vertical Warp Containment file (TarInc) file
283:    // IF the TarInc file could not be opened, THEN:
284:    //
285:    if( (pfVTarInc=fopen( sCmd_Line.pchVTarInc, "rb" )) == NULL )
286:    {
287:            //
288:            // Print an error message and abort
289:            //
290:            fprintf( stderr, "Could not open the Vertical Warp Containment file.\n");
291:            fprintf( stderr, "Please correct this problem and try again.\n");
292:            exit(0);
293:    //
294:    // END IF
295:    //
296:    }
297:    //
298:    // Read the Width and Height of the Vertical TarInc file
299:    //
300:    fread( &iVTarIncWidth, sizeof(UINT), 1, pfVTarInc );
301: fread( &iVTarIncHeight, sizeof(UINT), 1, pfVTarInc );
302: //
303: // Read the type of Vertical TarInc file
304: // (Used for future expansion)
305: //
306: fread( &iTarType, sizeof(UINT), 1, pfVTarInc );
307: //
308: // Allocate memory to hold one line of Vertical TarInc values
309: // IF the line could not be allocated, THEN:
310: //
311: if( (pVTarIncLine = (float *)malloc(sizeof(float)*iVTarIncWidth)) == NULL)
312: {
313:    //
314:    // Print error message and abort
315:    //
316:    fprintf( stderr, "Could not allocate memory for the Vertical TarInc line.\n");
317:    exit(1);
318: };
319: //
320:    // Open and load the input Targa File
321:    //
322:    TRY
323:    {
324:            SrcImage.Load( sCmd_Line.pchSrc );
325:    }
326:    CATCH( CException, theException )
327:    {
328:            TRACE( "Error while loading the Source Image.\n" );
329:            fcloseall();
330:            exit(1);
331:    }
332:    END_CATCH
333:    //
334:    // Verify that the width of the Containment file is 1 less
335:    // than the width of the source Targa file. This ensures
336:    // that only the spaces between the input Targa lines get
337:    // resized.
338:    //
339:    assert( iVTarIncWidth == SrcImage.Height() - 1 );
340:    //
341:    // Verify that both VTarInc and the Source image have the
342:    // same number of columns
343:    //
344:    assert( iVTarIncHeight == SrcImage.Width() );
345:    //
346:    // Print status message
347:    //
348:    fprintf( stderr, "Loading source image..[%s]\n", sCmd_Line.pchSrc );
349:    //
350:    // Allocate enough memory for the output Vertical Warp file
351:    //
```

```
TRY
{
        TarImage.Create( SrcImage.Width(), SrcImage.Height() );
}
CATCH( CException, theException )
{
        TRACE( "Error while allocating space for the Target Warp Image.\n" );
        fcloseall();
        exit(1);
}
END_CATCH
//
// LOOP for the number of vertical columns in the Containment Field
//
for( iColumn=0; iColumn < iVTarIncHeight; iColumn++)
{
        //
        // Read in a Vertical Tar Inc Line
        //
        fread( pVTarIncLine, sizeof(float), iVTarIncWidth, pfVTarInc );
        //
        // Read a source image column
        //
        pSrcCol = SrcImage.GetColumn(iColumn);
        //
        // Load the source image column into the VInterpolator
        //
        VInterpolator.SetSrc( pSrcCol );
        //
        // Get a target image column that will be loaded with
        // warped image pixels.
        //
        pTarCol = TarImage.GetColumn(iColumn);
        //
        // Initialize the VInterpolator and make copies of the
        // pointers to both the TarInc and Target image files
        //
        VInterpolator.Reset();
        VInterpolator.iTarIncWidth = iVTarIncWidth;
        pVTarIncLineTmp = pVTarIncLine;
        pTarColTmp = pTarCol;
        //
        // Print working line number
        //
        fprintf( stderr, "Working on column...[%d]\n", iColumn );
        //
        // Reset the Right Vertical MAC
        //
        VRightKernel.Reset();
        //
        // LOOP for the number of TarInc values in the Vertical Containment Field
        //
        for( iCount=0; iCount < iVTarIncWidth; iCount++ )
        {
                //
                // Reset the Left Vertical MAC
                //
                VLeftKernel.Reset();
                //
                // Calculate the correct Vertical Filterfactor based on TarInc
                //
                VFiltLookup.fTarInc = *pVTarIncLineTmp;
                //
                // Load the current VTarInc into the Vertical Upsample Increment Generator
                // Load the correct Vertical Filter Factor into the Upsample Increment Generator
                //
                VUpIncGen.fTarInc = *pVTarIncLineTmp;
                VUpIncGen.FiltFact = VFiltLookup.Out();
                //
                // Load the current TarInc into the VInterpolator
                // Load the Filter Factor into the VInterpolator
```

```
423:            // Load the Upsample Increment into the VInterpolator
424:            //
425:            VInterpolator.fTarInc = *pVTarIncLineTmp++;
426:            VInterpolator.FiltFact = VFiltLookup.Out();
427:            VInterpolator.UpInc = VUpIncGen.Out();
428:            //
429:            // Load the Filter Factor into the MACs
430:            //
431:            VLeftKernel.FiltFact = VFiltLookup.Out();
432:            VRightKernel.FiltFact = VFiltLookup.Out();
433:            //
434:            // IF the output pixel is a top edge pixel, THEN:
435:            //
436:            if( iCount==0 )
437:                    //
438:                    //          Double the Left filter kernel coefficients
439:                    //
440:                    VRightKernel.Double=TRUE;
441:            //
442:            // ELSE IF the output pixel is a bottom edge pixel, THEN:
443:            //
444:            else if( iCount==iVTarIncWidth-1 )
445:                    //
446:                    // Double the Right filter kernel coefficients
447:                    //
448:                    VLeftKernel.Double=TRUE;
449:            //
450:            // ELSE if the output pixel is somewhere in the middle of the image, THEN:
451:            //
452:            else
453:            {
454:                    //
455:                    // Do not double either Right or Left kernel
456:                    //
457:                    VLeftKernel.Double=FALSE;
458:                    VRightKernel.Double=FALSE;
459:            }
460:            //
461:            // END IF – output pixel is on an edge
462:            //
463:            //
464:            // Calculate the center tap Kernel values
465:            //
466:            VLeftKernel.MAC( VInterpolator.UpPix );
467:            VRightKernel.MAC( VInterpolator.UpPix );
468:            //
469:            // LOOP through the other FIR filter taps
470:            //
471:            while( VInterpolator.CenterTap() != TRUE )
472:            {
473:                    //
474:                    // Generate an interpolated pixel
475:                    // Filter the interpolated pixel
476:                    //
477:                    VUpPixel = VInterpolator.Generate();
478:                    VLeftKernel.MAC( VUpPixel );
479:                    VRightKernel.MAC( VUpPixel );
480:            //
481:            // END LOOP – other FIR filter taps
482:            //
483:            };
484:            //
485:            // Write the Right Kernel Output Pixel to the output line store
486:            //
487:            VFiltPix = VRightKernel.FilterPixel();
488:            *(pTarColTmp++) = VFiltPix.Red;
489:            *(pTarColTmp++) = VFiltPix.Grn;
490:            *(pTarColTmp++) = VFiltPix.Blu;
491:            //
492:            // Check to make sure that the Target Line stays within output image range
493:            //
```

```
494:                    assert( (UINT) (pTarColTmp-pTarCol-1) < TarImage.Width()*3 );
495:                    //
496:                    // Transfer the Left Kernel to the Right Kernel
497:                    //
498:                    VRightKernel.AccumRed=VLeftKernel.AccumRed;
499:                    VRightKernel.AccumGrn=VLeftKernel.AccumGrn;
500:                    VRightKernel.AccumBlu=VLeftKernel.AccumBlu;
501:            //
502:            // END LOOP – TarInc values from Containment File
503:            //
504:            }
505:            //
506:            // Write Left Kernel Output Pixel to the output line store
507:            //
508:            VFiltPix = VLeftKernel.FilterPixel();
509:            *(pTarColTmp++) = VFiltPix.Red;
510:            *(pTarColTmp++) = VFiltPix.Grn;
511:            *(pTarColTmp++) = VFiltPix.Blu;
512:            //
513:            // Check to make sure that the Target Line stays within range
514:            //
515:            assert( (UINT) (pTarColTmp-pTarCol-1) < TarImage.Width()*3 );
516:            //
517:            // Write the Output Pixel line to disk
518:            //
519:            TRY
520:            {
521:                    TarImage.PutColumn( pTarCol, iColumn );
522:            }
523:            CATCH( CException, theException )
524:            {
525:                    TRACE( "Error while writing a warped output image line.\n" );
526:                    fcloseall();
527:                    exit(1);
528:            }
529:            END_CATCH
530:    //
531:    // END LOOP – vertical columns
532:    //
533:    }
534:    //
535:    // Save vertically warped Targa file
536:    //
537:    TRY
538:    {
539:            TarImage.Save( "vert.tga" );
540:    }
541:    CATCH( CException, theException )
542:    {
543:            TRACE( "Error while saving the vertically warped image\n" );
544:            fcloseall();
545:            exit(1);
546:    }
547:    END_CATCH
548:    //
549:    // Close the Vertical Tar Inc file
550:    //
551:    fclose(pfVTarInc);
552:    //
553:    // Print status message
554:    //
555:    fprintf( stderr, "Finished Vertical Warping...\n");
556:    fprintf( stderr, "Starting Horizontal Warping...\n");
557:    //
558:    // ************************************************************************
559:    // Horizontal WARPING ALGORITHM
560:    // ************************************************************************
561:    //
562:    // Define all Horizontal warping variables
563:    //
564:    Pixel                   HUpPixel;                       // Horizontal Up sampled pixel
```

```
565:    Pixel           HFiltPix;           // Horizontal Filtered pixel
566:    Image           HSrcImage;          // Vertically Warped Source image to be
Horizontally warped
567:    Image           HTarImage;          // Output warped image
568:    LeftKernel      HLeftKernel;        // Horizontal Left FIR Filter Kernel MAC
569:    RightKernel     HRightKernel;       // Horizontal Right FIR Filter Kernel MAC
570:    FILE            *pfHTarInc;         // Pointer to Horizontal Warp TarInc
571:    UINT            iHTarIncWidth;      // Width of the the containment file
572:    UINT            iHTarIncHeight;     // Height of the containment file
573:    float           *pHTarIncLine;      // Pointer to a specific Horizontal warp containment (TarInc)
line
574:    float           *pHTarIncLineTmp;   // Copy of the previous pointer
575:    LOOKUP          HFiltLookup;        // Object to map the Horizontal TarInc to the correct FIR filter size
576:    IncGen          HUpIncGen;          // Object to generate a Horizontal Up Sample increment size
577:    Interpolate     HInterpolator;      // Object to generate an interpolated Horizontal up-sampled pixel
578:    BYTE            *pSrcRow;           // Pointer to a source image row
579:    BYTE            *pTarRow;           // Pointer to a target image row
580:    BYTE            *pTarRowTmp;        // Copy of the previous pointer
581:    UINT            iRow;               // Counts the number of rows in the
input image
582:    //
583:    // Open the Horizontal Warp Containment file (TarInc) file
584:    // IF the TarInc file could not be opened, THEN:
585:    //
586:    if( (pfHTarInc=fopen( sCmd_Line.pchHTarInc, "rb" )) == NULL )
587:    {
588:            //
589:            // Print an error message and abort
590:            //
591:            fprintf( stderr, "Could not open the Horizontal Warp Containment file.\n");
592:            fprintf( stderr, "Please correct this problem and try again.\n");
593:            exit(0);
594:    //
595:    // END IF
596:    //
597:    }
598:    //
599:    // Read the Width and Height of the Horizontal TarInc file
600:    //
601:    fread( &iHTarIncWidth, sizeof(UINT), 1, pfHTarInc );
602: fread( &iHTarIncHeight, sizeof(UINT), 1, pfHTarInc );
603: //
604: // Read the type of TarInc file
605: // (Used for future expansion)
606: //
607: fread( &iTarType, sizeof(UINT), 1, pfHTarInc );
608: //
609: // Allocate memory to hold one line of Horizontal TarInc values
610: // IF the line could not be allocated, THEN:
611: //
612: if( (pHTarIncLine = (float *)malloc(sizeof(float)*iHTarIncWidth)) == NULL)
613: {
614:    //
615:    // Print error message and abort
616:    //
617:    fprintf( stderr, "Could not allocate memory for the Horizontal TarInc line.\n");
618:    exit(1);
619: };
620: //
621:    // Open and load the Vertically warped Targa file
622:    //
623:    TRY
624:    {
625:            HSrcImage.Load( "vert.tga" );
626:    }
627:    CATCH( CException, theException )
628:    {
629:            TRACE( "Error while loading the Source Image.\n" );
630:            fcloseall();
631:            exit(1);
632:    }
```

```
633:    END_CATCH
634:    //
635:    // Verify that the width of the Containment file is 1 less
636:    // than the width of the source Targa file. This ensures
637:    // that only the spaces between the input Targa lines get
638:    // resized.
639:    //
640:    assert( iHTarIncWidth == HSrcImage.Width() - 1 );
641:    //
642:    // Verify that both HTarInc and the Source image have the
643:    // same number of lines
644:    //
645:    assert( iHTarIncHeight == HSrcImage.Height() );
646:    //
647:    // Allocate enough memory for the output Vertical Warp file
648:    //
649:    TRY
650:    {
651:            HTarImage.Create( HSrcImage.Width(), HSrcImage.Height() );
652:    }
653:    CATCH( CException, theExcoeption )
654:    {
655:            TRACE( "Error while allocating space for the Target Warp Image.\n" );
656:            fcloseall();
657:            exit(1);
658:    }
659:    END_CATCH
660:    //
661:    // LOOP for the number of Horizontal rows in the Containment Field
662:    //
663:    for( iRow=0; iRow < iHTarIncHeight; iRow++)
664:    {
665:            //
666:            // Read in a Horizontal Tar Inc Line
667:            //
668:            fread( pHTarIncLine, sizeof(float), iHTarIncWidth, pfHTarInc );
669:            //
670:            // Read a source image row
671:            //
672:            pSrcRow = HSrcImage.GetLine(iRow);
673:            //
674:            // Load the source image row into the HInterpolator
675:            //
676:            HInterpolator.SetSrc( pSrcRow );
677:            //
678:            // Get a target image row that will be loaded with
679:            // warped image pixels.
680:            //
681:            pTarRow = HTarImage.GetLine(iRow);
682:            //
683:            // Initialize the HInterpolator and make copies of the
684:            // pointers to both the TarInc and Target image files
685:            //
686:            HInterpolator.Reset();
687:            HInterpolator.iTarIncWidth = iHTarIncWidth;
688:            pHTarIncLineTmp = pHTarIncLine;
689:            pTarRowTmp = pTarRow;
690:            //
691:            // Print working line number
692:            //
693:            fprintf( stderr, "Working on row...[%d]\n", iRow );
694:            //
695:            // Reset the Right Horizontal MAC
696:            //
697:            HRightKernel.Reset();
698:            //
699:            // LOOP for the number of TarInc values in the Horizontal Containment Field
700:            //
701:            for( iCount=0; iCount < iHTarIncWidth; iCount++ )
702:            {
703:                    //
```

```
704:            // Reset the Left Horizontal MAC
705:            //
706:            HLeftKernel.Reset();
707:            //
708:            // Calculate the correct Horizontal Filterfactor based on TarInc
709:            //
710:            HFiltLookup.fTarInc = *pHTarIncLineTmp;
711:            //
712:            // Load the current HTarInc into the Horizontal Upsample Increment Generator
713:            // Load the correct Horizontal Filter Factor into the Upsample Increment Generator
714:            //
715:            HUpIncGen.fTarInc = *pHTarIncLineTmp;
716:            HUpIncGen.FiltFact = HFiltLookup.Out();
717:            //
718:            // Load the current TarInc into the HInterpolator
719:            // Load the Filter Factor into the HInterpolator
720:            // Load the Upsample Increment into the HInterpolator
721:            //
722:            HInterpolator.fTarInc = *pHTarIncLineTmp++;
723:            HInterpolator.FiltFact = HFiltLookup.Out();
724:            HInterpolator.UpInc = HUpIncGen.Out();
725:            //
726:            // Load the Filter Factor into the MACs
727:            //
728:            HLeftKernel.FiltFact = HFiltLookup.Out();
729:            HRightKernel.FiltFact = HFiltLookup.Out();
730:            //
731:            // IF the output pixel is a top edge pixel, THEN:
732:            //
733:            if( iCount==0 )
734:                    //
735:                    //          Double the Left filter kernel coefficients
736:                    //
737:                    HRightKernel.Double=TRUE;
738:            //
739:            // ELSE IF the output pixel is a bottom edge pixel, THEN:
740:            //
741:            else if( iCount==iHTarIncWidth-1 )
742:                    //
743:                    // Double the Right filter kernel coefficients
744:                    //
745:                    HLeftKernel.Double=TRUE;
746:            //
747:            // ELSE if the output pixel is somewhere in the middle of the image, THEN:
748:            //
749:            else
750:            {
751:                    //
752:                    // Do not double either Right or Left kernel
753:                    //
754:                    HLeftKernel.Double=FALSE;
755:                    HRightKernel.Double=FALSE;
756:            }
757:            //
758:            // END IF – output pixel is on an edge
759:            //
760:            //
761:            // Calculate the center tap Kernel values
762:            //
763:            HLeftKernel.MAC( HInterpolator.UpPix );
764:            HRightKernel.MAC( HInterpolator.UpPix );
765:            //
766:            // LOOP through the other FIR filter taps
767:            //
768:            while( HInterpolator.CenterTap() != TRUE )
769:            {
770:                    //
771:                    // Generate an interpolated pixel
772:                    // Filter the interpolated pixel
773:                    //
774:                    HUpPixel = HInterpolator.Generate();
```

```
                                HLeftKernel.MAC( HUpPixel );
                                HRightKernel.MAC( HUpPixel );
                        //
                        // END LOOP – other FIR filter taps
                        //
                        };
                        //
                        // Write the Right Kernel Output Pixel to the output line store
                        //
                        HFiltPix = HRightKernel.FilterPixel();
                        *(pTarRowTmp++) = HFiltPix.Red;
                        *(pTarRowTmp++) = HFiltPix.Grn;
                        *(pTarRowTmp++) = HFiltPix.Blu;
                        //
                        // Check to make sure that the Target Line stays within output image range
                        //
                        assert( (UINT) (pTarRowTmp-pTarRow-1) < HTarImage.Width()*3 );
                        //
                        // Transfer the Left Kernel to the Right Kernel
                        //
                        HRightKernel.AccumRed=HLeftKernel.AccumRed;
                        HRightKernel.AccumGrn=HLeftKernel.AccumGrn;
                        HRightKernel.AccumBlu=HLeftKernel.AccumBlu;
                //
                // END LOOP – TarInc values from Containment File
                //
                }
                //
                // Write Left Kernel Output Pixel to the output line store
                //
                HFiltPix = HLeftKernel.FilterPixel();
                *(pTarRowTmp++) = HFiltPix.Red;
                *(pTarRowTmp++) = HFiltPix.Grn;
                *(pTarRowTmp++) = HFiltPix.Blu;
                //
                // Check to make sure that the Target Line stays within range
                //
                assert( (UINT) (pTarRowTmp-pTarRow-1) < HTarImage.Width()*3 );
                //
                // Write the Output Pixel line to disk
                //
                TRY
                {
                        HTarImage.PutLine( pTarRow, iRow );
                }
                CATCH( CException, theException )
                {
                        TRACE( "Error while writing a warped output image line.\n" );
                        fcloseall();
                        exit(1);
                }
                END_CATCH
        //
        // END LOOP – horizontal lines
        //
        }
        //
        // Save warped Targa file
        //
        TRY
        {
                HTarImage.Save( sCmd_Line.pchWarp );
        }
        CATCH( CException, theException )
        {
                TRACE( "Error while saving the final Warp Image\n" );
                fcloseall();
                exit(1);
        }
        END_CATCH
        //
```

```
846:    // Free all allocated memory
847:    //
848:    free( pVTarIncLine );
849:    free( pHTarIncLine );
850:    SrcImage.FreeMemory();
851:    TarImage.FreeMemory();
852:    HSrcImage.FreeMemory();
853:    HTarImage.FreeMemory();
854:    fcloseall();
855:    //
856:    // Print status message
857:    //
858:    fprintf( stderr, "Program completed successfully!\n" );
859://
860:// END – Warping FAST!
861://
862:}
863:
```

```
1://
2:// Image Class – Encapsulates the Leadtools Image Functions
3://
4:// This image class greatly improves the LeadTools user interface by
5:// providing a simplified and consistent set of C++ member functions
6:// for manipulating images.
7://
8:// Program Name: IMAGE.CPP
9://
10:// Author: James D. E. Goel
11://
12://     - Initial release of the software. If you have any problems, please contact me.
13://
14:// Copyright (c)  1993 Genesis Microchip Inc.
15://                                                    200 Town Centre Blvd., Suite 400
16://                                                    Markham, Ontario
17://                                                    L3R 8G5
18://                                                    (905) 470-2742
19://
20://     All rights reserved.
21://
22:// Modifications:
23://
24://     Aug. 5, 1994          - James Goel             - Initial release
25://
26://
27:#include "image.h"
28:#include <stdlib.h>
29:#include <assert.h>
30:#include <afx.h>
31:
32://
33:// Error codes
34://
35:#define NAME_SIZE              1
36:#define SAVE_ERR               2
37:#define READ_ERR               3
38:#define MEM_ERR                4
39:
40://
41:// ****************************************************************************
42:// error -- Prints appropriate error message and aborts gracefully.
43:// ****************************************************************************
44://
45:void Image::Error( int errnum )
46:{
47:     //
48:     // SWITCH ON: Associated error
49:     //
50:     switch(errnum)
51:     {
52:               //
53:               // ON CASE: Command line parameter error
54:               //
55:    case NAME_SIZE:
56:     //
57:     // Print out a file name size error message
58:     //
59:                    fprintf ( stderr, "ERROR: One of the input file names exceeds the legal 8.3 file name size.\n");
60:                    fprintf ( stderr, "Please reduce the size of the file name and try again.\n" );
61:               //
62:               // END CASE: command line
63:               //
64:               break;
65:               //
66:               // ON CASE: Leadtools could not save the output image
67:               //
68:               case SAVE_ERR:
69:                    //
70:                    // Print out save error message
71:                    //
```

```
72:                     fprintf( stderr, "ERROR: Could not save the output image file.\n");
73:                     fprintf( stderr, "The disk may be full or corrupted. ");
74:                     fprintf( stderr, "Please correct these conditions and try again.\n");
75:             //
76:             // END CASE: save error.
77:             //
78:             break;
79:             //
80:             // ON CASE: Could not read the input image
81:             //
82:             case READ_ERR:
83:                     //
84:                     // Print out load error message
85:                     // Stop execution
86:                     //
87:                     fprintf( stderr, "ERROR: Could not read one of the the input files.\n");
88:                     fprintf( stderr, "The disk may be corrupted.");
89:                     fprintf( stderr, "Please correct these conditions and try again.\n");
90:             //
91:             // END CASE: load error
92:             //
93:             break;
94:             //
95:             // ON CASE: Could not allocate the required memory
96:             //
97:             case MEM_ERR:
98:                     //
99:                     // Print out error
100:                    //
101:                    fprintf( stderr, "ERROR: Could not allocate the required memory.\n");
102:                    fprintf( stderr, "Please free up some DOS memory and try again.\n");
103:            //
104:            // END CASE: memory error
105:            //
106:            break;
107:            //
108:            // ON DEFAULT: A fatal internal error has been generated
109:            //
110:    default:
111:    //
112:    // Print out fatal internal error
113:    // Stop execution
114:    //
115:                    printf("ERROR: A fatal internal error has occurred. Please call James Goel.\n");
116:            //
117:            // END DEFAULT: fatal error
118:            //
119:            break;
120:    //
121:    // END SWITCH: associated error
122:    //
123:    }
124://
125:// END ERROR
126://
127:}
128:
129://
130:// Default Constructor
131://
132:Image::Image(void)
133:{
134:     FlagAllocLine = FALSE;
135:     FlagAllocCol = FALSE;
136://
137:// END Default Constructor
138://
139:}
140:
141://
142:// Default Destructor
```

```
143://
144:Image::~Image(void)
145:{
146:    TRACE( "Freeing Bitmap...\n" );
147:    L_FreeBitmap( &ImageBitmap );
148:    delete pLine;
149://
150:// END Default Destructor
151://
152:}
153:
154://
155:// Free Bitmaps
156://
157:void Image::FreeMemory(void)
158:{
159:    TRACE( "Freeing Bitmap and memory...\n" );
160:    L_FreeBitmap( &ImageBitmap );
161:    delete pLine;
162:    delete pCol;
163:}
164:
165:// ************************
166:// Construct an empty image
167:// ************************
168:void Image::Create( UINT iImgWidth, UINT iImgHeight )
169:{
170:    CFileException *theException = new CFileException;
171:    //
172:    // Initialize the bitmap to new width and height
173:    // IF the bitmap was NOT initialized successfully, THEN:
174:    //
175:    L_InitBitmap( &ImageBitmap, iImgWidth, iImgHeight, 24);
176:    //
177:    // Allocate memory
178:    // IF the memory cannot be allocated, THEN:
179:    //
180:    if( L_AllocateBitmap( &ImageBitmap, TYPE_CONV ) != SUCCESS )
181:    {
182:            //
183:            // Print error message and abort
184:            //
185:            Error( MEM_ERR );
186:            L_FreeBitmap( &ImageBitmap );
187:            delete pLine;
188:            THROW( theException );
189:    //
190:    // END IF -- Allocate the bitmap memory
191:    //
192:    }
193:    iWidth = iImgWidth;
194:    iHeight = iImgHeight;
195:    //
196:    // Initialize some basic Bitmap properties
197:    //
198:    ImageBitmap.ViewPerspective = BOTTOM_LEFT;
199:    ImageBitmap.Order = ORDER_BGR;
200://
201:// END -- Construct and empty image
202://
203:}
204:// ************************
205:// Load in an Image File
206:// ************************
207:void Image::Load( char *pchInput )
208:{
209:    CFileException *theException = new CFileException;
210:    //
211:    // Load Targa Image
212:    // IF the image could not be loaded, THEN:
213:    //
```

```
214:    if( L_LoadTGAScreen( pchInput, &ImageBitmap, OUTPUT_BITMAP, TYPE_CONV, 0, 0)!= SUCCESS )
215:    {
216:            //
217:            // Print error message and abort
218:            //
219:            Error( READ_ERR );
220:            THROW( theException );
221:    //
222:    // END IF -- load Targa image
223:    //
224:    }
225:    //
226:    // Copy the width and height information to the local private variables
227:    //
228: iWidth = ImageBitmap.Width;
229: iHeight = ImageBitmap.Height;
230:// END -- Load image file
231://
232:}
233:// ********************
234:// Save an Image File
235:// ********************
236:void Image::Save( char *pchOutput )
237:{
238:    CFileException *theException = new CFileException;
239:    //
240:    // Save Targa Image
241:    // IF the image could not be saved, THEN:
242:    //
243:    if( L_SaveTGABitmap( pchOutput, &ImageBitmap, 24)!= SUCCESS )
244:    {
245:            //
246:            // Print error message and abort
247:            //
248:            Error( SAVE_ERR );
249:            L_FreeBitmap( &ImageBitmap );
250:            delete pLine;
251:            THROW( theException );
252:    //
253:    // END IF -- Save Targa image
254:    //
255:    }
256://
257:// END -- Save image file
258://
259:}
260:// ************************
261:// Return the Image Width
262:// ************************
263:UINT Image::Width( void )
264:{
265:    return( iWidth);
266://
267:// END -- return image width
268://
269:}
270:// ************************
271:// Return the Image Height
272:// ************************
273:UINT Image::Height( void )
274:{
275:    return( iHeight);
276://
277:// END -- return image Height
278://
279:}
280:// ***************************
281:// Get a line from the Image
282:// ***************************
283:BYTE *Image::GetLine( UINT iLineNum )
284:{
```

```
285:    CFileException *theException = new CFileException;
286:    assert( iLineNum < (UINT) ImageBitmap.Height );
287:    //
288:    // IF a line has not been allocated, THEN:
289:    //
290:    if( FlagAllocLine == FALSE )
291:    {
292:            //
293:            // Allocate the Line
294:            //
295:            pLine = new BYTE[ImageBitmap.BytesPerLine];
296:            FlagAllocLine = TRUE;
297:    //
298:    // END IF -- a line has been allocated
299:    //
300:    }
301:    //
302:    // IF the origin is in the bottom left corner, THEN:
303:    //
304:    if( ImageBitmap.ViewPerspective == BOTTOM_LEFT )
305:    {
306:            // Adjust the row number based on the origin of the image
307:            iLineNum = ImageBitmap.Height - iLineNum - 1;
308:    //
309:    // END IF -- origin in bottom-left corner
310:    //
311:    }
312:    //
313:    // Load the allocated line with data from the image
314:    // IF the line could not be loaded, THEN:
315:    //
316:    if( L_GetBitmapRow( &ImageBitmap, pLine, iLineNum, ImageBitmap.BytesPerLine)<0)
317:    {
318:            //
319:            // Print Error and abort
320:            //
321:            Error (MEM_ERR);
322:            L_FreeBitmap( &ImageBitmap );
323:            delete pLine;
324:            THROW( theException );
325:    //
326:    // END IF -- line could not be loaded
327:    //
328:    }
329:    return( pLine );
330://
331:// END -- get a line from the Image
332://
333:}
334:// ***************************
335:// Put a line in the Image
336:// ***************************
337:void Image::PutLine( BYTE *pImgLine, UINT iLineNum )
338:{
339:    CFileException *theException = new CFileException;
340:    assert( iLineNum < (UINT) ImageBitmap.Height );
341:    //
342:    // IF the origin is in the bottom left corner, THEN:
343:    //
344:    if( ImageBitmap.ViewPerspective == BOTTOM_LEFT )
345:    {
346:            // Adjust the row number based on the origin of the image
347:            iLineNum = ImageBitmap.Height - iLineNum - 1;
348:    //
349:    // END IF -- origin in bottom-left corner
350:    //
351:    }
352:    //
353:    // Put the line into the image
354:    // IF the line could not be placed, THEN:
355:    //
```

```
356:    if( L_PutBitmapRow( &ImageBitmap, pImgLine, iLineNum, ImageBitmap.BytesPerLine)<0)
357:    {
358:            //
359:            // Print Error and abort
360:            //
361:            Error (MEM_ERR);
362:            L_FreeBitmap( &ImageBitmap );
363:            delete pLine;
364:            THROW( theException );
365:    //
366:    // END IF – line could not be placed
367:    //
368:    }
369://
370:// END – put a line in the Image
371://
372:}
373:// ****************************
374:// Get a column from the Image
375:// ****************************
376:BYTE *Image::GetColumn( UINT iColNum )
377:{
378:    UINT    iRowNum;
379:    UINT    iRow;
380:    BYTE    pColPixel[3];
381:    BYTE    *pColTmp;
382:    assert( iColNum < (UINT) ImageBitmap.Width );
383:    //
384:    // IF a column has not been allocated, THEN:
385:    //
386:    if( FlagAllocCol == FALSE )
387:    {
388:            //
389:            // Allocate the Column
390:            //
391:            printf( "Allocating a column space..\n");
392:            pCol = new BYTE[ImageBitmap.Height*3];
393:            FlagAllocCol = TRUE;
394:    //
395:    // END IF – a column has been allocated
396:    //
397:    }
398:    pColTmp = pCol;
399:    //
400:    // LOOP through the number of pixels in a specific column
401:    //
402:    for( iRowNum=0; iRowNum < (UINT) ImageBitmap.Height; iRowNum++ )
403:    {
404:            iRow = iRowNum;
405:            //
406:            // IF the origin is in the bottom left corner, THEN:
407:            //
408:            if( ImageBitmap.ViewPerspective == BOTTOM_LEFT )
409:            {
410:                    // Adjust the row number based on the origin of the image
411:                    iRow = ImageBitmap.Height - 1 - iRowNum ;
412:            //
413:            // END IF – origin in bottom-left corner
414:            //
415:            }
416:            //
417:            // Read RGB pixel from the specific column
418:            //
419:            L_GetBitmapRowCol( &ImageBitmap, pColPixel, iRow, iColNum, 3 );
420:            //
421:            // Copy the RGB pixel to the output buffer
422:            assert( pColTmp-pCol < ImageBitmap.Height*3 );
423:            *(pColTmp++) = *pColPixel;
424:            *(pColTmp++) = *(pColPixel+1);
425:            *(pColTmp++) = *(pColPixel+2);
426:    //
```

```
427:    // END LOOP – number of pixels in a specific column
428:    //
429:    }
430:    return( pCol );
431://
432:// END – get a column from the Image
433://
434:}
435://
436:// ****************************
437:// Put a column in the Image
438:// ****************************
439://
440:void Image::PutColumn( BYTE* pCol, UINT iColNum )
441:{
442:    UINT    iRowNum;
443:    UINT    iRow;
444:    BYTE    *pColTmp;
445:    BYTE    *pLineTmp;
446:    assert( iColNum < (UINT) ImageBitmap.Width );
447:    assert( pCol != NULL );
448:    //
449:    // Create a temporary line holder
450:    //
451:    pLineTmp = new BYTE[ImageBitmap.BytesPerLine];
452:    //
453:    // Copy the original column pointer
454:    //
455:    pColTmp = pCol;
456:    //
457:    // LOOP through the number of pixels in a specific column
458:    //
459:    for( iRowNum=0; iRowNum < (UINT) ImageBitmap.Height; iRowNum++ )
460:    {
461:            iRow = iRowNum;
462:            //
463:            // IF the origin is in the bottom left corner, THEN:
464:            //
465:            if( ImageBitmap.ViewPerspective == BOTTOM_LEFT )
466:            {
467:                    // Adjust the row number based on the origin of the image
468:                    iRow = ImageBitmap.Height - 1 - iRowNum ;
469:            //
470:            // END IF – origin in bottom-left corner
471:            //
472:            }
473:            //
474:            // Copy the RGB pixel to the output buffer
475:            //
476:            assert( pColTmp-pCol < ImageBitmap.Height*3 );
477:            //
478:            // Write the RGB pixel to the specific column
479:            //
480:            L_GetBitmapRow( &ImageBitmap, pLineTmp, iRow, ImageBitmap.BytesPerLine );
481:            pLineTmp[0+iColNum*3] = *pColTmp++;
482:            pLineTmp[1+iColNum*3] = *pColTmp++;
483:            pLineTmp[2+iColNum*3] = *pColTmp++;
484:            L_PutBitmapRow( &ImageBitmap, pLineTmp, iRow, ImageBitmap.BytesPerLine );
485:            //
486:            // NOTE : I suspect a bug in this Leadtools routine
487:            //
488:            //L_PutBitmapRowCol( &ImageBitmap, ColPixel, iRow, iColNum, 3 );
489:    //
490:    // END LOOP – number of pixels in a specific column
491:    //
492:    }
493:    delete pLineTmp;
494://
495:// END – put a column in the Image
496://
497:}
```

```
1://
2:// Genesis Logic Classes 1.0
3://
4:// This set of classes implements some basic logic functions
5:// for use in the Genesis Warping Engine software.
6://
7:// Program Name: LOGIC.CPP
8://
9:// Author: James D. E. Goel
10://
11://     - Initial release of the software. If you have any problems, please contact me.
12://
13:// Copyright (c)  1993 Genesis Microchip Inc.                    200 Town Centre Blvd., Suite 400
14://                                                               Markham, Ontario
15://                                                               L3R 8G5
16://                                                               (905) 470-2742
17://
18://
19://     All rights reserved.
20://
21:// Modifications:
22://
23://     Aug. 8, 1994          - James Goel
24://                                                               - Initial release
25://
26:
27:#include "logic.h"
28:#include <assert.h>
29:#include <stdio.h>
30:#include <afx.h>
31://
32:// Create some useful type definitions
33://
34:typedef unsigned char BYTE;
35:typedef unsigned char PIXEL;
36:typedef unsigned long ULONG;
37:typedef unsigned int UINT;
38:
39:// ************************************************************************
40:// D-Flop Register Class
41:// ************************************************************************
42://
43:// This class simulates a logical DFLOP.
44://
45:// ****************************************
46:// D-Flop Clock transfers inputs to outputs
47:// ****************************************
48:void REG::Clk( void )
49:{
50:     Out = In;
51://
52:// END -- Constructor
53:}
54:
55:// ************************************************************************
56:// Filter Lookup Table
57:// ************************************************************************
58://
59:// This class simulates the Filter Lookup table in the warping
60:// patent. (reproduced below for completeness)
61://
62:// TarInc        0<q<=1    1<q<=2    2<q<=4    4<q<=8    8<q<=16   16<q<=32
63:// FiltSize       3 Tap     5 Tap     9 Tap     17 Tap    33 Tap    65 Tap
64:// FiltFactor     0         1         2         3         4         5
65://
66:// 1. TarInc is the space between output Target pixels.
67:// 2. FiltSize is the FIR filter tap size
68:// 3. FiltFactor is the number of right shifts required to divide the TarInc
69:// into an appropriate Up Sample Increment.
70://
71:// Constructor and Destructor
```

```
72://
73:LOOKUP::LOOKUP( void )
74:{
75:};
76:LOOKUP::~LOOKUP( void )
77:{
78:};
79://-------------------------------------
80:// Out – Outputs a filter factor based on the fTarInc value
81://-------------------------------------
82:UINT LOOKUP::Out( void )
83:{
84:     assert( fTarInc > 0 );
85:     //
86:     // IF TarInc is in the first category, THEN:
87:     //
88:     if( (fTarInc > 0) && (fTarInc <= 1))
89:               //
90:               // Return a Filter Factor of 0 for the 3 Tap FIR filter
91:               //
92:               return 0;
93:     //
94:     // ELSE IF TarInc is in the second category, THEN:
95:     //
96:     else if ( (fTarInc > 1) && (fTarInc <= 2))
97:               //
98:               // Return a Filter Factor of 1 for the 5 Tap FIR filter
99:               //
100:              return 1;
101:    //
102:    // ELSE IF TarInc is in the third category, THEN:
103:    //
104:    else if ( (fTarInc > 2) && (fTarInc <= 4))
105:              //
106:              // Return a Filter Factor of 2 for the 9 Tap FIR filter
107:              //
108:              return 2;
109:    //
110:    // ELSE IF TarInc is in the fourth category, THEN:
111:    //
112:    else if ( (fTarInc > 4) && (fTarInc <= 8))
113:              //
114:              // Return a Filter Factor of 3 for the 17 Tap FIR filter
115:              //
116:              return 3;
117:    //
118:    // ELSE IF TarInc is in the fifth category, THEN:
119:    //
120:    else if ( (fTarInc > 8) && (fTarInc <= 16))
121:              //
122:              // Return a Filter Factor of 4 for the 33 Tap FIR filter
123:              //
124:              return 4;
125:    //
126:    // ELSE IF TarInc is in the sixth category, THEN:
127:    //
128:    else if ( (fTarInc > 16) && (fTarInc <= 32))
129:              //
130:              // Return a Filter Factor of 5 for the 65 Tap FIR filter
131:              //
132:              return 5;
133:    //
134:    // ELSE TarInc does not fit into any category, THEN:
135:    //
136:    else
137:    {
138:              //
139:              // Print internal error message
140:              //
141:              fprintf( stderr, "Fatal internal error.\n");
142:              printf( "fTarInc...[%f]\n", fTarInc );
```

```
143:                return 5;
144:    //
145:    // END IF – Filter Lookup Table
146:    //
147:    };
148://
149:// END FUNCTION– LOOKUP::Out
150://
151:};
152:
153:// ************************
154:// Three Input Comparator
155:// ************************
156://
157:// Compares three inputs and outputs the largest one.
158://
159:UINT TRICOMP::Out( void )
160:{
161:    UINT    iTemp;
162:    //
163:    // Output the largest of three values
164:    //
165:    if( A > B)
166:                iTemp = A;
167:    else
168:                iTemp = B;
169:
170:    if( iTemp > C)
171:                return iTemp;
172:    else
173:                return C;
174://
175:// END FUNCTION – TRICOMP
176://
177:};
178:
179:// ************************
180:// Two Input Comparator
181:// ************************
182://
183:// Compares two inputs and outputs the largest one.
184://
185:UINT BICOMP::Out( void )
186:{
187:    //
188:    // Output the largest of two values
189:    //
190:    if( A > B )
191:                return A;
192:    else
193:                return B;
194://
195:// END FUNCTION – BICOMP
196://
197:};
198:
199:// ********************************
200:// Upsample Increment Generator
201:// ********************************
202://
203:// Right shifts Tarinc by the Filter Factor supplied by the
204:// Filter Lookup table.
205://
206:ULONG IncGen::Out( void )
207:{
208:    ULONG   lUpInc;
209:    //
210:    // Convert fTarInc to an integer value for
211:    // faster and easier arthimetic calculations
212:    //
213:    lUpInc = (ULONG) ( fTarInc * INDEX_PRECISION_VALUE);
```

```
214:      return ( lUpInc >> FiltFact );
215://
216:// END FUNCTION -- Out
217://
218:};
219:
220:// **************
221:// Interpolator
222:// **************
223://
224:// Generates interpolated pixels based on the size of the Filter Factor
225:// Generates a flag that indicates it has come to the end of a TarInc
226:// value.
227:// ---------------------------
228:// Default Constructor and Destructor
229:// ---------------------------
230:Interpolate::Interpolate( void )
231:{
232:      //
233:      // Initialize private variables
234:      //
235:      Accum = 0L;
236:      UpIncCount = 0;
237:      NextPixPos = 0L;
238:      CurrPixPos = 0L;
239://
240:// END -- Interpolate::Constructor
241://
242:};
243:Interpolate::~Interpolate( void )
244:{
245:};
246:// ---------------------------------------
247:// Reset -- Reset and initialize the interpolator
248:// ---------------------------------------
249:void Interpolate::Reset(void)
250:{
251:      //
252:      // Initialize private variables
253:      //
254:      Accum = 0L;
255:      UpIncCount = 0;
256:      NextPixPos = 0L;
257:      CurrPixPos = 0L;
258:      iTarIncWidth = 0;
259:      //
260:      // Advance the current pixel value to be the next pixel value
261:      //
262:      SrcCurrPix.Red = *pSrcLine++;
263:      SrcCurrPix.Grn = *pSrcLine++;
264:      SrcCurrPix.Blu = *pSrcLine++;
265:      //
266:      // Load SrcNextPix = GetNewSrc(pix)
267:      //
268:      SrcNextPix.Red = *pSrcLine++;
269:      SrcNextPix.Grn = *pSrcLine++;
270:      SrcNextPix.Blu = *pSrcLine++;
271:      //
272:      // Initialize the Upsampled Pixel
273:      //
274:      UpPix.Red=SrcCurrPix.Red;
275:      UpPix.Grn=SrcCurrPix.Grn;
276:      UpPix.Blu=SrcCurrPix.Blu;
277://
278:// END -- Interpolate::Reset
279://
280:};
281:// ---------------------------------
282:// SetSrcLine -- Function to set the Source Line
283:// ---------------------------------
284:void Interpolate::SetSrc( BYTE *pLine )
```

```
285:{
286:    //
287:    // Load the source line into a private pointer
288:    //
289:    pSrcLine = pLine;
290:    pSrcLineTmp = pLine;
291://
292:// END -- Interpolate::SetSrcLine
293://
294:};
295:// ---------------------------------------------------
296:// CenterTap -- Indicates when an interpolated pixel is the center tap for an
297://          FIR filter
298:// ---------------------------------------------------
299:BOOLEAN Interpolate::CenterTap( void )
300:{
301:    //
302:    // IF the interpolator is creating the first upsampled pixel OR
303:    // IF the interpolator is creating the last upsampled pixel THEN
304:    //
305:    if( UpIncCount*2 == (UINT) ( 2 << FiltFact) )
306:    {
307:            //
308:            // Initialize the UpSample increment counter and return a TRUE
309:            //
310:            UpIncCount = 0;
311:            return TRUE;
312:    }
313:    else
314:    {
315:            //
316:            // Advance the UpIncCount
317:            //
318:            UpIncCount++;
319:            return FALSE;
320:    }
321:    //
322:    // END IF -- interpolator
323:    //
324://
325:// END -- Interpolator::CenterTap
326://
327:};
328:
329:// ---------------------
330:// Out -- Generates and up-sample pixel
331:// ---------------------
332:Pixel Interpolate::Generate( void )
333:{
334:    ULONG Test;
335:    //
336:    // Accumulate the Up Sampled Increment
337:    //
338:    Accum += UpInc;
339:    //
340:    // The next pixel position is derived from the integer portion of accumulated UpInc
341:    //
342:    NextPixPos = GetInt( Accum );
343:    //
344:    // IF Next pixel position requires a new source pixel, AND
345:    // IF the next pixel has not gone past the end of the TarInc line, THEN:
346:    //
347:    if( (NextPixPos > CurrPixPos) && (NextPixPos < iTarIncWidth) )
348:    {
349:            //
350:            // Advance the current pixel position
351:            //
352:            CurrPixPos = NextPixPos;
353:            //
354:            // Advance the current pixel value to be the next pixel value
355:            //
```

```
356:                SrcCurrPix.Red = SrcNextPix.Red;
357:                SrcCurrPix.Grn = SrcNextPix.Grn;
358:                SrcCurrPix.Blu = SrcNextPix.Blu;
359:                //
360:                // Load SrcNextPix = GetNewSrc(pix)
361:                //
362:                SrcNextPix.Red = *pSrcLine++;
363:                SrcNextPix.Grn = *pSrcLine++;
364:                SrcNextPix.Blu = *pSrcLine++;
365:    //
366:    // END IF – NextPixPos > CurrPixPos
367:    //
368:    };
369:    //
370:    // Calculate interpolated pixel using linear interpolation.
371:    // UpPix = SrcCurrPix + Fractional(Accum) * (SrcNextPix - SrcCurrPix )
372:    //
373:    Test = GetFrac(Accum);
374:    UpPix.Red = (BYTE) SrcCurrPix.Red + (BYTE)
375:                                        (GetFrac(Accum)*(SrcNextPix.Red - SrcCurrPix.Red)>>INDEX_PRECISION);
376:    UpPix.Grn = (BYTE) SrcCurrPix.Grn + (BYTE)
377:                                        (GetFrac(Accum)*(SrcNextPix.Grn - SrcCurrPix.Grn)>>INDEX_PRECISION);
378:    UpPix.Blu = (BYTE) SrcCurrPix.Blu + (BYTE)
379:                                        (GetFrac(Accum)*(SrcNextPix.Blu - SrcCurrPix.Blu)>>INDEX_PRECISION);
380:    return UpPix;
381://
382:// END FUNCTION – Interpolator::Out
383://
384:};
385:// ************************
386:// Left FIR Filter Kernel
387:// ************************
388://
389:// This class creates the left half of an Output Pixel based
390:// on the FIR filter indicated by the Filter Factor.
391://
392:// Default Constructor and Destructor
393://
394:LeftKernel::LeftKernel( void )
395:{
396:};
397:LeftKernel::~LeftKernel( void )
398:{
399:};
400:// ---------------------------
401:// Reset – Initializes the Left Kernel MAC
402:// ---------------------------
403:void LeftKernel::Reset( void )
404:{
405:    //
406:    // Initialize private variables
407:    //
408:    Double=FALSE;
409:    KernelPos=0;
410:    AccumRed=0;
411:    AccumGrn=0;
412:    AccumBlu=0;
413://
414:// END – LeftKernel::Reset
415://
416:};
417:// ---------------
418:// MAC – Filter a pixel
419:// ---------------
420:void LeftKernel::MAC( Pixel UpPix )
421:{
422:    UINT iEdgeFactor = 1;
423:    //
424:    // IF the kernel is doubled, THEN:
425:    //
426:    if( Double==TRUE )
```

```
427:                    //
428:                    // Double the UpSampled Pixel
429:                    //
430:                    iEdgeFactor = 2;
431:    //
432:    // END IF -- double the kernel
433:    //
434:    // Accumulate the FIR Coefficient * UpSampled Pixel
435:    //
436:    AccumRed += iLeftCoeff[FiltFact][KernelPos] * UpPix.Red * iEdgeFactor;
437:    AccumGrn += iLeftCoeff[FiltFact][KernelPos] * UpPix.Grn * iEdgeFactor;
438:    AccumBlu += iLeftCoeff[FiltFact][KernelPos] * UpPix.Blu * iEdgeFactor;
439:    //
440:    // Advance to the next kernel position
441:    //
442:    KernelPos++;
443:    //
444:    // IF the kernel has rolled over, THEN:
445:    //
446:    if( KernelPos > (UINT) (1 << FiltFact))
447:                    //
448:                    // Initialize the Kernel
449:                    //
450:                    KernelPos=0;
451:    //
452:    // END IF -- kernel has rolled over
453:    //
454://
455:// END -- LeftKernel::MAC
456://
457:};
458:// ------------------------
459:// FilterPixel -- Return a Filter Pixel
460:// ------------------------
461:Pixel LeftKernel::FilterPixel( void )
462:{
463:    FiltPix.Red = (BYTE) (AccumRed >> UBYTE_MAX);
464:    FiltPix.Blu = (BYTE) (AccumBlu >> UBYTE_MAX);
465:    FiltPix.Grn = (BYTE) (AccumGrn >> UBYTE_MAX);
466:    //
467:    // Clamp the Filtered pixel if required
468:    //
469:    if( AccumRed < 0)
470:                    FiltPix.Red = (BYTE) 0;
471:    if( AccumGrn < 0)
472:                    FiltPix.Grn = (BYTE) 0;
473:    if( AccumBlu < 0)
474:                    FiltPix.Blu = (BYTE) 0;
475:
476:    if( AccumRed > 65280 )
477:                    FiltPix.Red = (BYTE) 255;
478:    if( AccumGrn > 65280 )
479:                    FiltPix.Grn = (BYTE) 255;
480:    if( AccumBlu > 65280 )
481:                    FiltPix.Blu = (BYTE) 255;
482:
483:    return( FiltPix );
484://
485:// END -- LeftKernel::FilterPixel
486://
487:};
488:// ************************
489:// Right FIR Filter Kernel
490:// ************************
491://
492:// This class creates the Right half of an Output Pixel based
493:// on the FIR filter indicated by the Filter Factor.
494://
495:// Default Constructor and Destructor
496://
497:RightKernel::RightKernel( void )
```

```
498:{
499:};
500:RightKernel::~RightKernel( void )
501:{
502:};
503:// ---------------------------
504:// Reset -- Initialize the Right Kernel MAC
505:// ---------------------------
506:void RightKernel::Reset( void )
507:{
508:     //
509:     // Initialize private variables
510:     //
511:     Double = FALSE;
512:     KernelPos=0;
513:     AccumRed=0;
514:     AccumGrn=0;
515:     AccumBlu=0;
516://
517:// END -- RightKernel::Reset
518://
519:};
520:// ---------------
521:// MAC -- Filter a pixel
522:// ---------------
523:void RightKernel::MAC( Pixel UpPix )
524:{
525:     UINT iEdgeFactor = 1;
526:     //
527:     // IF the kernel is doubled, THEN:
528:     //
529:     if( Double==TRUE )
530:               //
531:               // Double the UpSampled Pixel
532:               //
533:               iEdgeFactor = 2;
534:     //
535:     // END IF -- double the kernel
536:     //
537:     // Accumulate the FIR Coefficient * UpSampled Pixel
538:     //
539:     AccumRed += iRightCoeff[FiltFact][KernelPos] * UpPix.Red * iEdgeFactor;
540:     AccumGrn += iRightCoeff[FiltFact][KernelPos] * UpPix.Grn * iEdgeFactor;
541:     AccumBlu += iRightCoeff[FiltFact][KernelPos] * UpPix.Blu * iEdgeFactor;
542:     KernelPos++;
543:     //
544:     // IF the kernel has rolled over, THEN:
545:     //
546:     if( KernelPos > (UINT) (1 << FiltFact))
547:               //
548:               // Initialize the Kernel
549:               //
550:               KernelPos=0;
551:     //
552:     // END IF -- kernel has rolled over
553:     //
554://
555:// END -- RightKernel::MAC
556://
557:};
558:// ---------------------------
559:// FilterPixel -- Return a Filter Pixel
560:// ---------------------------
561:Pixel RightKernel::FilterPixel( void )
562:{
563:     //
564:     // Convert the accumulated filtered pixel to a byte
565:     //
566:     FiltPix.Red = (BYTE) (AccumRed >> UBYTE_MAX);
567:     FiltPix.Grn = (BYTE) (AccumGrn >> UBYTE_MAX);
568:     FiltPix.Blu = (BYTE) (AccumBlu >> UBYTE_MAX);
```

```
569:    //
570:    // Clamp the Filtered pixel if required
571:    //
572:    if( AccumRed < 0)
573:            FiltPix.Red = (BYTE) 0;
574:    if( AccumGrn < 0)
575:            FiltPix.Grn = (BYTE) 0;
576:    if( AccumBlu < 0)
577:            FiltPix.Blu = (BYTE) 0;
578:
579:    if( AccumRed > 65280 )
580:            FiltPix.Red = (BYTE) 255;
581:    if( AccumGrn > 65280 )
582:            FiltPix.Grn = (BYTE) 255;
583:    if( AccumBlu > 65280 )
584:            FiltPix.Blu = (BYTE) 255;
585:    return( FiltPix );
586://
587:// END – RightKernel::FilterPixel
588://
589:};
```

```
1://
2:// Genesis Logic Classes 1.0
3://
4:// This set of classes implements some basic logic functions
5:// for use in the Genesis Warping Engine software.
6://
7:// Program Name: LOGIC.CPP
8://
9:// Author: James D. E. Goel
10://
11://      - Initial release of the software. If you have any problems, please contact me.
12://
13:// Copyright (c)  1993 Genesis Microchip Inc.
14://                                                    200 Town Centre Blvd., Suite 400
15://                                                    Markham, Ontario
16://                                                    L3R 8G5
17://                                                    (905) 470-2742
18://
19://      All rights reserved.
20://
21:// Modifications:
22://
23://      Aug. 8, 1994          - James Goel
24://                                                    - Initial release
25://
26:
27:#ifndef __LOGICCLASS_H__
28:#define __LOGICCLASS_H__
29:#endif
30:
31:#ifndef __cplusplus
32:#error Microsoft Foundation Classes require C++ compilation (use a .cpp suffix)
33:#endif
34:
35:typedef unsigned char BYTE;
36:typedef unsigned char PIXEL;
37:typedef unsigned long ULONG;
38:typedef unsigned int UINT;
39:
40://
41:// Define a Pixel structure
42://
43:typedef struct pixel_tag
44:{
45:      BYTE      Red;
46:      BYTE      Grn;
47:      BYTE      Blu;
48:} Pixel;
49:
50://
51:// Setup a boolean type with TRUE=1 and FALSE=0
52://
53:typedef unsigned int BOOLEAN;
54:#define TRUE 1
55:#define FALSE 0
56:
57://
58:// Define the fractional precision of TarInc
59://
60:#define INDEX_PRECISION                    16L              // 16 fractional bits of precision
61:#define INDEX_PRECISION_VALUE        (1L << INDEX_PRECISION)
62:#define INDEX_PRECISION_MASK  ((1L << INDEX_PRECISION) - 1L)
63:
64://
65:// Define the number of bits per byte
66://
67:#define UBYTE_MAX 8
68:
69:#define GetInt(Value)              ((ULONG)((Value) >> INDEX_PRECISION))
70:#define GetFrac(Value)             ((Value) & INDEX_PRECISION_MASK)
71:
```

```
72://
73:// gm865x1 FIR Filter Coefficients
74://
75:#define MAX_FILTER                6          // Maximum filter type number
76:#define MAX_TAPS                  33         // Maximum number of filter taps (Kernal doubled)
77:
78://
79:// Left Kernel Coefficients
80://
81:const long iLeftCoeff[MAX_FILTER][MAX_TAPS] =
82:{
83:     { 0, 128, 0,  0,  0,  0,  0,  0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
84:     {-16, 64, 80, 0,  0,  0,  0,  0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
85:     { -7,  3, 30, 63, 39, 0,  0,  0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
86:     { -3, -2,  1,  7, 15, 23, 31, 37, 19, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
87:     { -1, -1, -1,  0,  0,  2,  3,  5,  7, 9,11,13,15,16,33,24, 9, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
88:     {  1,  1,  1,  2,  2,  3,  3,  4,  4, 5, 5, 6, 6, 7, 7, 8, 8, 8, 8,  9, 9, 16, 5, 0, 0, 0, 0, 0, 0},
89:};
90://
91:// Right Kernel Coefficients
92://
93:const long iRightCoeff[MAX_FILTER][MAX_TAPS] =
94:{
95:     {128, 0,  0,  0,  0,  0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
96:     {80, 64, -16, 0,  0,  0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
97:     {39, 63, 30,  3, -7,  0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
98:     {19, 37, 31, 23, 15,  7,  1,-2,-3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
99:     {9,  24, 33, 16, 15, 13, 11, 9, 7, 5, 3, 2, 0, 0,-1,-1,-1, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
100:    {5,  16,  9,  9,  8,  8,  8, 8, 7, 7, 6, 6, 5, 5, 4, 4, 3, 3, 2,  2, 1, 1, 1, 0, 0, 0, 0, 0, 0},
101:};
102:
103:// ********************************************************************************
104:// D-Flop Register Class
105:// ********************************************************************************
106://
107:// This class simulates a logical DFLOP.
108://
109:class REG
110:{
111:private:
112:public:
113:     //
114:     // Input and Output Register Value
115:     //
116:     UINT      In;
117:     UINT      Out;
118:     //
119:     // Class Constructor and Destructor
120:     //
121:     REG( void );
122:     ~REG( void );
123:     //
124:     // Clock register value
125:     //
126:     void Clk( void );
127://
128:// END CLASS - REG
129://
130:};
131:
132:// ********************************************************************************
133:// Filter Lookup Table
134:// ********************************************************************************
135://
136:// This class simulates the Filter Lookup table in the warping
137:// patent. (reproduced below for completeness)
138://
139:// TarInc       0<a<=1    1<a<=2    2<a<=4    4<a<=8    8<a<=16   16<a<=32
140:// FiltSize     3 Tap     5 Tap     9 Tap     17 Tap    33 Tap    65 Tap
141:// FiltFactor    0         1         2         3         4         5
142://
```

```
143:// 1. TarInc is the space between output Target pixels.
144:// 2. FiltSize is the FIR filter tap size
145:// 3. FiltFactor is the number of right shifts required to divide the TarInc
146:// into an appropriate Up Sample Increment.
147://
148:class LOOKUP
149:{
150:private:
151:public:
152:     //
153:     // Constructor and Destructor
154:     //
155:     LOOKUP(void);
156:     ~LOOKUP(void);
157:     //
158:     // Output Target pixel spacing
159:     //
160:     float     fTarInc;
161:     //
162:     // Mapped Filter Factor for the input TarInc
163:     //
164:     UINT Out(void);
165:};
166:
167:// ************************
168:// Three Input Comparator
169:// ************************
170://
171:// Compares three inputs and outputs the largest one.
172://
173:class TRICOMP
174:{
175:private:
176:public:
177:     //
178:     // Constructor and Destructor
179:     //
180:     TRICOMP(void);
181:     ~TRICOMP(void);
182:     //
183:     // Three input variables
184:     //
185:     UINT A,B,C;
186:     //
187:     // Generate the output
188:     //
189:     UINT Out(void);
190:};
191:
192:// ************************
193:// Two Input Comparator
194:// ************************
195://
196:// Compares two inputs and outputs the largest one.
197://
198:class BICOMP
199:{
200:private:
201:public:
202:     //
203:     // Constructor and Destructor
204:     //
205:     BICOMP(void);
206:     ~BICOMP(void);
207:     //
208:     // Two input variables
209:     //
210:     UINT A,B;
211:     //
212:     // Generate the output
213:     //
```

```
214:     UINT Out(void);
215:};
216:
217:// ******************************
218:// Upsample Increment Generator
219:// ******************************
220://
221:// Right shifts TarInc by the Filter Factor supplied by the
222:// Filter Lookup table.
223://
224:class IncGen
225:{
226:private:
227:public:
228:     //
229:     // Upsample Filter Factor
230:     //
231:     UINT      FiltFact;
232:     float     fTarInc;
233:     //
234:     // Output an Upsample Increment
235:     //
236:     ULONG Out( void );
237:};
238:// *************
239:// Interpolator
240:// *************
241://
242:// Generates Interpolated pixels based on the size of the Filter Factor
243:// Generates a flag that indicates it has come to the end of a TarInc
244:// value.
245://
246:class Interpolate
247:{
248:private:
249:     UINT      UpIncCount;
250:     ULONG     CurrPixPos, NextPixPos;
251:     Pixel     SrcCurrPix,SrcNextPix;
252:     BYTE      *pSrcLine, *pSrcLineTmp;
253:public:
254:     //
255:     // Upsample Filter Factor
256:     //
257:     ULONG     Accum;
258:     UINT      FiltFact;
259:     float     fTarInc;
260:     Pixel     UpPix;
261:     BYTE      bPixel;
262:     ULONG     UpInc;
263:     UINT      iTarIncWidth;
264:     //
265:     // Constructor and Destructor
266:     //
267:     Interpolate( void );
268:     ~Interpolate( void );
269:     //
270:     // Resets the interpolator to a known state
271:     //
272:     void Reset( void );
273:     //
274:     // Function to set the pointer to the Source Line
275:     //
276:     void SetSrc( BYTE *pLine );
277:     //
278:     // Generate an Interpolated Pixel
279:     //
280:     Pixel Generate( void );
281:     //
282:     // Flag indicates when the center kernel
283:     // tap interpolated pixel is being generated
284:     //
```

```
285:      BOOLEAN CenterTap( void );
286:};
287:// ****************************
288:// Left FIR Filter Kernel MAC
289:// ****************************
290://
291:// This class creates the left half of an Output Pixel based
292:// on the FIR filter indicated by the Filter Factor.
293://
294:class LeftKernel
295:{
296:private:
297:     UINT      KernelPos;
298:public:
299:     long      AccumRed;
300:     long      AccumGrn;
301:     long      AccumBlu;
302:     BOOLEAN        Double;
303:     Pixel     FiltPix;
304:     UINT      FiltFact;
305:     //
306:     // Constructor and Destructor
307:     //
308:     LeftKernel( void );
309:     ~LeftKernel( void);
310:     //
311:     // Reset MAC to a known state
312:     //
313:     void Reset( void );
314:     //
315:     // Generate a filtered pixel
316:     //
317:     void MAC( Pixel UpPix );
318:     //
319:     // Return a Filtered Pixel
320:     //
321:     Pixel FilterPixel( void );
322://
323:// END CLASS – LeftKernelMAC
324://
325:};
326:// ****************************
327:// Right FIR Filter Kernel MAC
328:// ****************************
329://
330:// This class creates the right half of an Output Pixel based
331:// on the FIR filter indicated by the Filter Factor.
332://
333:class RightKernel
334:{
335:private:
336:     UINT      KernelPos;
337:public:
338:     long      AccumRed;
339:     long      AccumGrn;
340:     long      AccumBlu;
341:     BOOLEAN        Double;
342:     Pixel     FiltPix;
343:     UINT      FiltFact;
344:     //
345:     // Constructor and Destructor
346:     //
347:     RightKernel( void );
348:     ~RightKernel( void);
349:     //
350:     // Reset MAC to a known state
351:     //
352:     void Reset( void );
353:     //
354:     // Generate a filtered pixel
355:     //
```

```
356:      void MAC( Pixel UpPix );
357:      //
358:      // Returns a Filtered Pixel
359:      //
360:      Pixel FilterPixel( void );
361://
362:// END CLASS – RightKernelMAC
363://
364:};
```

```
1://
2:// Image Class - Encapsulates the Leadtools Image Functions
3://
4:// This image class greatly improves the LeadTools user interface by
5:// providing a simplified and consistent set of C++ member functions
6:// for manipulating images.
7://
8:// Program Name: IMAGE.CPP
9://
10:// Author: James D. E. Goel
11://
12://      - Initial release of the software. If you have any problems, please contact me.
13://
14:// Copyright (c)  1993 Genesis Microchip Inc.
15://                                                    200 Town Centre Blvd., Suite 400
16://                                                    Markham, Ontario
17://                                                    L3R 8G5
18://                                                    (905) 470-2742
19://
20://     All rights reserved.
21://
22:// Modifications:
23://
24://     Aug. 5, 1994          - James Goel
25://                                                    - Initial release
26://
27:// Class definition
28://
29://
30:#define FOR_DOS
31:#include "l_toolap.h"
32:#include "l_bitmap.h"
33:#include "l_error.h"
34://
35:// Setup LEAD TOOLS for DOS definitions
36://
37:#ifndef __IMAGECLASS_H__
38:#define __IMAGECLASS_H__
39:#endif
40:
41:#ifndef __cplusplus
42:#error Microsoft Foundation Classes require C++ compilation (use a .cpp suffix)
43:#endif
44:
45:typedef unsigned char BYTE;
46:typedef unsigned char PIXEL;
47:typedef unsigned long ULONG;
48:typedef unsigned int UINT;
49:typedef unsigned int BOOLEAN;
50:
51:#define TRUE 1
52:#define FALSE 0
53:
54:class Image
55:{
56:private:
57:     BITMAPHANDLE    ImageBitmap;
58:     char       pchInputFilename[13];
59:     char       pchOutputFilename[13];
60:     UINT       iWidth;
61:     UINT       iHeight;
62:     UINT       iLine;
63:     BYTE       *pLine;
64:     BYTE       *pCol;
65:     BYTE       *pTGALine;
66:     FILE       *pfTarga;
67:     BOOLEAN         FlagAllocLine;
68:     BOOLEAN         FlagAllocCol;
69:     void       Error( int errnum );
70:
71:public:
```

```
72:     //
73:     // Class Constructor and Destructor
74:     //
75:     Image( void );
76:     ~Image( void );
77:     //
78:     // Free allocated Bitmap and memory
79:     //
80:     void FreeMemory(void);
81:     //
82:     // Load and Save image file
83:     //
84:     void Load( char *pchInput );
85:     void Save( char *pchOutput );
86:     //
87:     // Get image Width and Height
88:     //
89:     UINT Width( void );
90:     UINT Height( void );
91:     //
92:     // Get and Put image lines
93:     //
94:     BYTE *GetLine( UINT iLineNum );
95:     void PutLine( BYTE *pLine, UINT iLineNum );
96:     //
97:     // Get and Put image columns
98:     //
99:     BYTE *GetColumn( UINT iLineNum );
100:    void PutColumn( BYTE *pLine, UINT iLineNum );
101:    //
102:    // Create an empty image
103:    //
104:    void Create( UINT iWidth, UINT iHeight );
105://
106:// END CLASS – Image
107://
108:};
```

【図１】

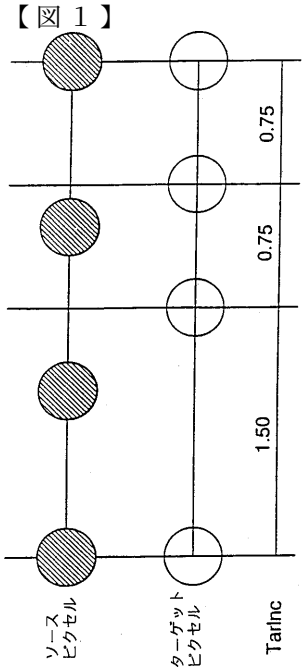


FIG.1

【図２】

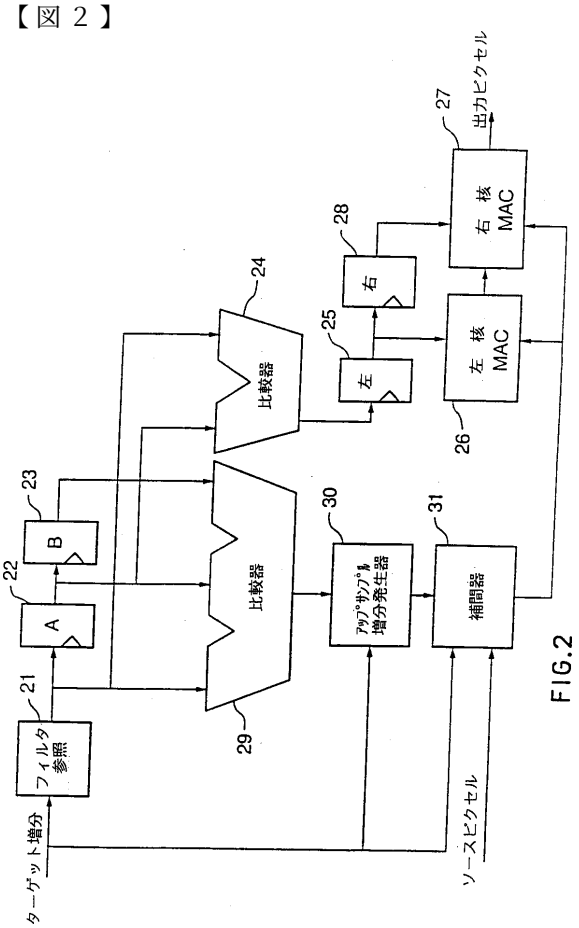


FIG.2

【図３】

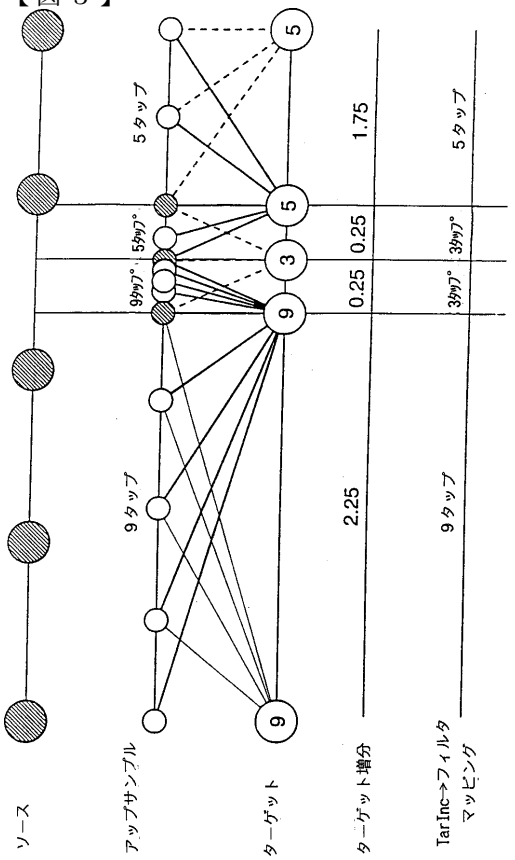


FIG.3

【図４】

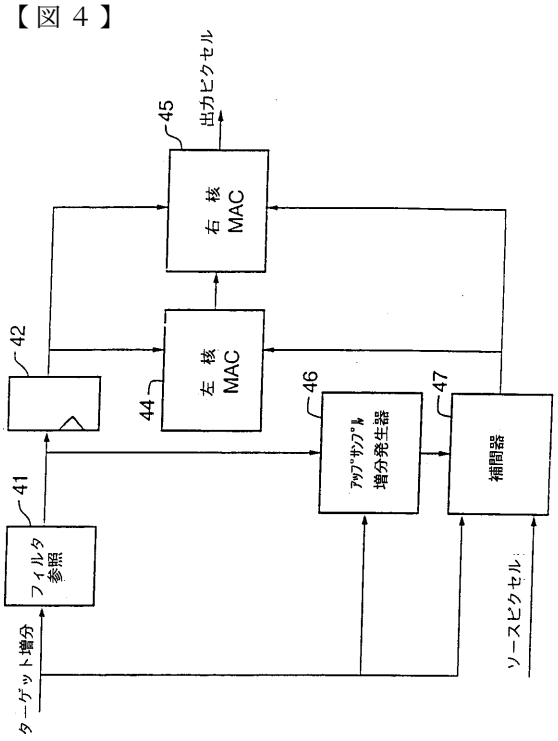


FIG.4

【図５】

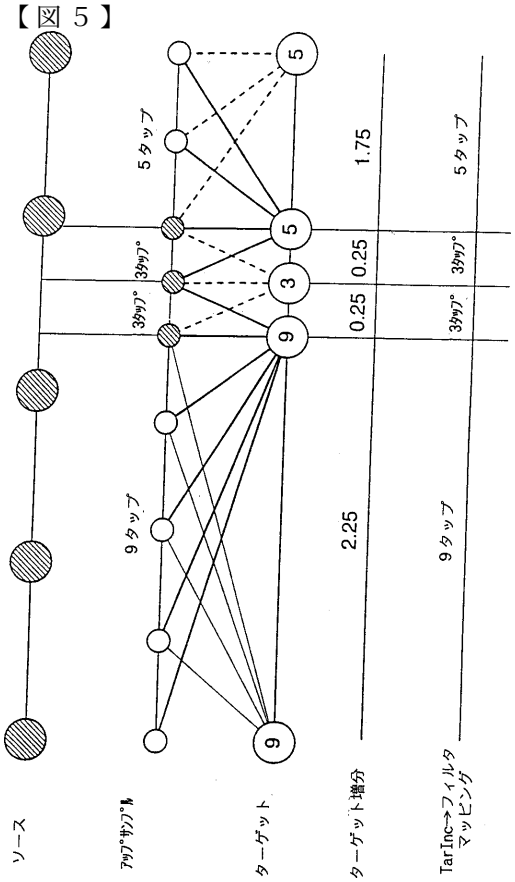


FIG.5

フロントページの続き

(72)発明者　ゴエル,　ジェームス
カナダ　オンタリオ　Ｌ１Ｓ　７Ｈ８　アハックス　レイクワット　ドライヴ　ウエスト　１９３　スウィート　４１５
(72)発明者　バイヤー,　ロバート
カナダ　オンタリオ　Ｎ２Ｔ　２Ｊ２　ワーテルロー　ティンバークロフト　クレッセント　４３８

審査官　▲徳▼田　賢二

(56)参考文献　特開平８－１０７５５８（ＪＰ，Ａ）
特開平８－３７６６４（ＪＰ，Ａ）
特表平９－５０５９１４（ＪＰ，Ａ）
米国特許第５３５５３２８（ＵＳ，Ａ）
特開平３－２０６７０６（ＪＰ，Ａ）
特開平７－２１３６０（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.^{7}，ＤＢ名)
H04N　5/262 - 5/28
G06T　3/40