Jan-Mar 2018 #5

ATARI 400/800/XL/XE

EXCEL

Paul Lay Interview

Game Maps:
Jack the Nipper,
Spellbinder & Jetset Willy

Retro Programming an
80s Atari Game

Video 61 Cartridge Games

Guide to the Best
Modern Atari
8-bit Peripherals

EXCEL

ISSUE 5 CONTENTS

Many thanks to:
AtariAge.com / Atarimania.Com / Mapy.Atari.info
Avery Lee for the Altirra Atari Emulator
All Atari 8-Bit Coders, Artists & Musicians
David Sherwin / Robin Edwards
Jonathan Halliday / Matthew Baker
Michael Walters / Lukas Bezdek
Excel magazine is a non-profit publication. All archive material remains the copyright of the original owners and is used in accordance with UK fair usage laws.

This and previous issues are also available as downloadable PDFs at:

WEBSITE: WWW.EXCEL-RETRO-MAG.CO.UK

EDATARIAL by Robert Stuart

Good grief! Is it really eight months since the last issue? How time flies! I really wanted to get this issue out for Christmas 2017 but events conspired against me and here it is, two months late - but the extra time has enabled reviews of some great new games. Once again, as in the last issue, we have two new games conversions from Mariusz - Skool Daze and Jack the Nipper (with Tezz) and a review of the long awaited and eagerly anticipated Bosconian!

Also in this issue: we have a round-up of hot new hardware for the Atari from Robin (Electrotrains) Edwards, some great new maps courtesy of Lukas Bezdek including the new games Spellbinder and Jack the Nipper and an interview with top Atari programmer Paul Lay courtesy of Jonathan (Flashjazzcat) Halliday. From the other side of the pond, we have reviews of some Video 61 cartridge games by Matt (Gunstar) Baker and part two of Adventures in Atari Programming by Michael (WEBmikey) Walters which will conclude next issue. David Sherwin from Canada has provided a review of the Polish Collectors Edition Set 1 together with a short chat with the man behind them: Jurek Dudek.

Hopefully by next issue (Summer... maybe...?) we will have some superb looking new games in the shape of Barbarian, Hobgoblin 2, Time Pilot and 8 Bit Slicks. We also plan on reviewing more cartridge-only games from Video 61 including Venture, Xenophobe, Helicommander, Tempest Extreem and the ABBUC entry Rain of Terror. Another ABBUC entry, Jason Kendall's game Space Fortress Omega will shortly be available to order on disk (see Coming Soon page...)

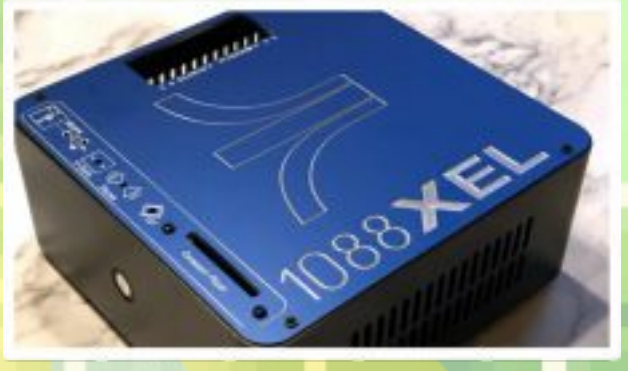
As well as the Tezz interview, we'll be taking a look at his other work, including Chimera+ (with a map) and other Atari games outwith the ones already reviewed (Manic Miner, Saboteur, Bomb Jack and Jack the Nipper) in previous issues of the magazine, which includes his re-working of the Jocky Wilson Darts Compendium and the Boulderdash 25th Anniversary Edition.

Next year (November 2019 to be precise) will be the fortieth anniversary of the release of the original

Atari 400 and 800 computers, so I have a few ideas about doing something to celebrate this. A calendar will definitely be designed, and I have a couple of ideas for Atari 8-bit books which have been floating around my head for a few months... stay tuned!

Some other ideas for forthcoming issues are a look at the new mini-ITX Atari 1088XEL, which is a fantastic modern hardware product designed to continue the lifespan of our favourite 8-bit well into the future, with support for all the latest technologies such as USB ports, DVI sockets and SD Cards. Atari 8-bit software emulation will also be covered including the amazing Altirra for Windows, the Colleen and Droid800 Android emulators and the Atari XLbox for the X-Box.

Robert



Quite a lot of diehard Atari users think that they were short changed back in the day. We knew our machine had great graphic and sound specs for the time... but so many times the Atari version of any given game was inferior to that on other computers. Worse (depending on your point of view), some games were never converted to the Atari at all. Bosconian is just such a game. Originally released into the frenzied arcades of 1981 by Namco in Japan, and manufactured and distributed in the US by Midway, this was quite a revolutionary arcade game for the time. One of the first (if not the actual first) shoot 'em ups with a scrolling background, it was also one of the first games to offer a "continue" feature whereby you could pump more coins into the machine and continue playing from where you left off. It also used a landscape screen when most of the early arcade games of the time used a portrait mode. And it had digitised speech!

The arcade game utilises similar hardware to the earlier Galaga game and indeed, many Bosconian machines were subsequently converted into Galaga machines which was a more popular and lucrative title - perhaps because playing Bosconian can be quite a frustrating experience. Despite this, it won the 1982 Arcade Award for Best Science Fiction / Fantasy Coin-Op Game in January 1983, beating the top contenders - Atari's Gravitar and Sega's Zaxxon.

Home computer conversions were thin on the ground, other than the pretty faithful MSX version in 1984, but in 1987 Mastertronic, in a very rare foray into releasing licensed arcade games, belatedly brought out home conversions called

Bosconian 87 for the Sinclair ZX Spectrum, Amstrad CPC and Commodore 64. You might have thought that after paying for the license they would have released an Atari version to maximise their investment, especially since the conversion was handled by Binary Design, who produced Amaurote, Storm and Feud for the Atari at around the same time, but... nope! However...

Fast forward a mere 30 years and... wow! The extremely talented Polish coder Janusz Chabowski (shanti77 on Atari Age) released his final version to the Atari 8-bit community and surpassed everyone's expectations with an absolutely first class Atari conversion, with all the features of the original game, including the speech.

The music and sound effects by Michal Szpilowski are terrific and the superbly drawn title screen was created by Krzysztof Ziembik. There are actually several variants of this Atari version, which include the standard 64k and enhanced (with speech) 128k versions for both NTSC and PAL systems. And best of all, they are free to download.

If Mastertronic had produced an Atari conversion in 1987, we might not have been gifted with this little miracle - especially since none of the other home versions include the digitised speech and in any case, a 128k version (imagine loading that from a cassette) would have been nothing but a pipe dream at the time.

The title page differs on the 64 and 128k versions although the excellent music is retained in both versions. The 128k version starts with a digitised burst of “Blast off!” and other speech in the game includes "Alert! Alert!" (enemies approaching), "Battle stations!" (formation attack), "Spy ship sighted!" (Spy ship advancing), "Condition red!" (enemy attacks become more aggressive which occurs if you take too long to clear a round, or if you miss the spy ship). You may actually prefer the 64k version as the speech gets a bit repetitive after a while. Every time you lose a life you are treated to another “Blast off!” and once you run out of ships, you’ll hear “Game over!” The 64k versions uses warning sound effects instead of the speech when alerting you to impending trouble.

Your aim is to score as many points as possible by destroying the enemy space stations. You control a small ship (complete with animated engine jets) that rotates in the middle of the screen, which scrolls in all directions around you. An original feature here is that you fire your laser cannon both forwards and backwards a the same time, which comes in very handy! Each round consists of a number of huge green enemy space station bases that must be destroyed to advance to the next round. On the right of the screen is a radar map to assist with the base locations and a report of the current alert status, which begins at yellow. Each station consists of six cannons arranged in a hexagon, surrounding a central red core. You can destroy all six cannons (200 points each) or shoot the core itself to destroy the station with one shot (1,500 points), but after the first round, the core starts defending itself by opening and closing while launching missiles and it is only vulnerable to your laser whilst it is open.

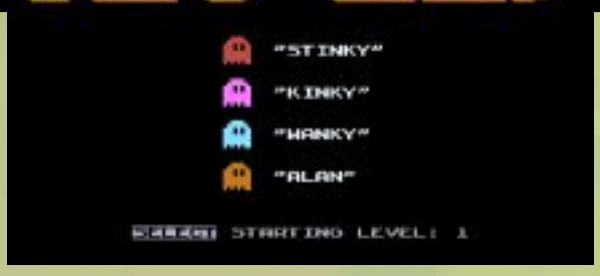
You must also avoid (or better still, destroy) the stationary asteroids (10 points) and mines (20 points), which otherwise tend to get in your way. A variety of enemy missiles and kamikaze fighters will attempt to collide with your ship and a good strategy is to turn away from them and blast them with your rear cannon. Enemy bases will also occasionally launch a squadron of ships in formation attacks - destroying the leader causes the remaining enemies to disperse, but destroying all enemies in a formation will score you extra points.

A spy ship (worth a random bonus value) will also appear occasionally, which must be destroyed or the round will go to "Condition red" regardless of how long the current round has taken. In addition, the enemy fighter pilots aren’t very well trained and they do occasionally crash into their own mines - which helps you out a bit! The game plays like a dream - it’s fast and furious, but does take a little time to get used to the control method and twin-laser firing ship. It definitely has the “one more go” factor - it is very, very addictive! Apparently there are 22 levels to negotiate, but I haven’t managed to get past level 5 yet.

The in-game graphics are very colourful, with silky smooth scrolling, and the joystick response is superb. Interestingly (from a programmer’s viewpoint), the Atari’s hardware sprites are only used for your own ship and the entire status bar at the right side of the screen - the scores, ships remaining and current alert status.

Everything here is top notch - the presentation is superb and the graphics and music are superb. So, after all those years, Atari users can now boast of the best 8-bit conversion of Bosconian! This is definitely a contender for the Atari 8-bit game of the year.

PAC-MAD



Well, who would have thought that of all the games in all the world, the Atari 8-bit needed another Pac Man clone? Wojciech "Bocianu" Bociański, that's who. He has programmed this new version of the classic game in MadPascal (hence the name Pac Mad, which took me a while to realise... doh!), with nice musical ditties from LiSU. The game was introduced at the third Ironia party in Wysoka, Poland in August 2017 and finished in second place to "Tensor Trzaskowskiego" which I haven't had time to play properly yet, so it'll be reviewed next issue. You can blame Bosconian for that!

So, does this game offer anything different from the official Pac Man games? Well, for a start, the ghosts move faster in this one - a lot faster! These buggers can really move, so you have to really be on your toes to keep out of their way. The Power Pellets slow them down but even when slowed, they are still about the speed of the ghosts in regular Pac Man, so a little more welly is required to catch and eat them. As usual you have to gobble up all the Pac Dots to clear each level, but the levels get bigger and have to be scrolled up or down to get at the other dots. The playfields vary in size and the bigger ones scroll up and down vertically. Some of the mazes use quite unusual designs and an online screen editor is available to create and test your own maze layouts.

The graphics are perfunctory, with your Pac person and all the ghosts being simple single-colour sprites and the backdrops are nicely enough drawn in several shades of colour. As in the original game, random fruits appear on each level which can gain you extra points and shortcuts are available to skip from one edge of the screen to another. If you get a high score, a DataMatrix picture is displayed, which allows you to send your score to the High Score Cafe website via a smartphone, as shown On the right.

The game is slick, nicely presented and gameplay is fast and furious. The graphics are good and the sound is too. Is it addictive? Hell, yes! Definitely a case of "just one more go... and then I'll wash the dishes..."

SKOOL DAZE

Skool Daze, published by Microsphere in 1984 is an early Sinclair ZX Spectrum game and is fondly remembered, coming in at No. 4 in a list of the all time best Spectrum games by Your Sinclair in 2004. It was a big seller and was quickly converted to the Commodore 64, on which this Atari version was based. Once again, the port comes from Mariusz Wojcieszek with the original Spectrum graphics fine-tuned for the Atari by Jose Pereira. It's great to see another old classic from the Spectrum which uses the Atari hi-res mode coloured with hardware sprites, as used to great effect in Saboteur and especially in Manic Miner.

The default game stars a schoolboy called Eric (although you can rename him) and the aim of the game is to steal your report card from a safe in the school staff room which is, curiously, achieved by solving various puzzles in the game. The computer controls all the other characters. There are four teachers: Mr Wacker (the headmaster), Mr Rockitt (the science teacher), Mr Withit (the geography teacher) and Mr Creak (the history master). Only three of the pupils are named: Boy Wonder (the tearaway), Angelface (the bully) and Einstein (the swot). You can rename any or all of these characters before starting the game but the rest of the pupils remain nameless. The characters communicate with the use of comic strip speech bubbles.

During the game, the status bar at the bottom centre of the screen will tell you which lesson you are supposed to be attending, or if it's playtime or dinnertime. If you are caught out of class or otherwise misbehaving, a nearby teacher will punish you by issuing lines - and sometimes quite a lot of lines! When 10,000 lines or more are accumulated, the game ends with your expulsion from school. You can receive lines for all sorts of infringements, such as sitting on the floor or simply being the nearest pupil to a teacher who has been hit by a catapult fired by another pupil, so you have to try and prevent other pupils from getting you into trouble. Your boy has his own catapult and is also capable of throwing a good punch, but it's best not to indulge in violence or tomfoolery if there's a teacher around. Also, keep out of the Staff Room when there are teachers in it, or it's more lines for you!

You will notice coloured shields on the walls dotted around the school. These all have to be struck by Eric or hit with a catapult shot (deflected off a teacher's noggin once you've floored him with an initial shot) and this is the first part of the puzzle to solve - the shields will flash once activated. What the shields have to do with extricating your report card from the staff room is anyone's guess. You can also use other pupils to help reach the higher placed shields: knock them out and then use the prostrate heap as a trampoline! This may sound a bit harsh, but then the other kids tend to get you into all sorts of trouble by fighting, writing on the blackboard (and blaming you) and generally grassing you to the teachers, so go ahead - wallop them!

Playing Skool Daze involves a bit of keyboard dexterity - the various actions all involve pressing letters on your keyboard - i.e. "H" to hit, "J" for jump or "S" to stand (if you've been knocked down or are seated) and some of the classes have a shortage of seats so you have to resort to knocking another pupil out of their seat or you end up getting more lines for not sitting down. There is a fair bit of running around involved as the playing area is relatively small and some rooms are walled off from each other so you often have to go the long way round.

The game uses a flip screen method to draw the rooms, (as in the C64) while the Spectrum original used juddery scrolling. The sound effects are basic bleeps ported from the C64 and there is a short tune on the title page. One plus on the Atari is that it runs faster than both the C64 and Spectrum versions and it is also better looking than the unofficial ports for the Oric (remember that?) and especially the Amstrad version, which looks ghastly in shades of pink. This is yet another minor miracle for the old Atari 8-bit! The school report should say: "Terrific work! Keep it up!"

JACK THE NIPPER

Hot on the heels of Skool Daze comes another Spectrum conversion featuring a troublesome kid - although this one is still in nappies (or diapers, if you're in the US). For all you comics fans, Jack the Nipper is loosely based on Leo Baxendale's British comic strip character Sweeny Toddler, who first appeared in Shiver and Shake in 1973. The game was published by Gremlin Graphics in 1986 for the ZX Spectrum, Commodore 64, Amstrad CPC and for the MSX, which was almost exactly the same as the Spectrum version. As is usually the case, there was no Atari version, although Gremlin Graphics did produce Trailblazer and Footballer of the Year for the Atari in the same year, before dropping support for the machine the following year after porting Basil the Mouse Detective. Jack the Nipper was a big success, scoring highly in all the popular magazines, reaching #2 in the sales charts and kept off the top spot by Elite's superb Ghosts and Goblins.

Mariusz Wojcieszek (busy man - two game ports this issue) and Terrence Derby (Tezz) are responsible for this Atari version, which is also faithful to the Spectrum version, with Benn Daglish's music and sound effects converted from the Amstrad. A lot of time was spent rewriting the sprite / screen engine for a significant increase in gameplay performance compared to the other platform releases. The game program automatically detects the extra RAM of a 130XE and uses the additional memory for a pre-shifted / mirrored software sprite cache which gives a further performance boost. As you can see on the screenshots opposite, the Amstrad used it's hi-res (320x200) mode in four colours and the C64 version uses a lower resolution (160x200) with more colour (and multicoloured sprites), which does look rather nice compared to all the other versions and seems to be a bit faster as well.

Jack the Nipper is a side-view flip screen game with puzzle solving and platform elements. The graphics are rendered in 2D with an illusion of depth achieved by the characters moving forward and back (or more accurately, up and down) within the playfield. High resolution mode has been utilised on the Atari with hardware sprites being used to provide extra colour. The characters are nicely drawn and animated, although they do tend to wander around the screens like headless chickens. There are various human characters, who don't seem to mind sharing their space with two-legged dogs, ghosts, a living sock and blobby ectoplasm things. There are around fifty different screens to explore.

The loading screen is nicely done, again in hi-res mode with sprite overlays to add the colour and the title page once the game loads is equally colourful and has a nice little tune. Other in game controls are “H” to pause the game and “M” to toggle the music on or off.

The aim of the game is to achieve a rating of “Little Horror” with a naughtiness level of 100%, which is measured as a percentage on the handy Naughtyometer under the playing area, together with your number of lives and an “irritant diaper dermatitis” meter - which measures your Nappy Rash level. Contact with the other characters will result in a good spanking for Jack, which increases your rash level(!), costing you a life when it reaches maximum. Ouch! You start the game with five lives.

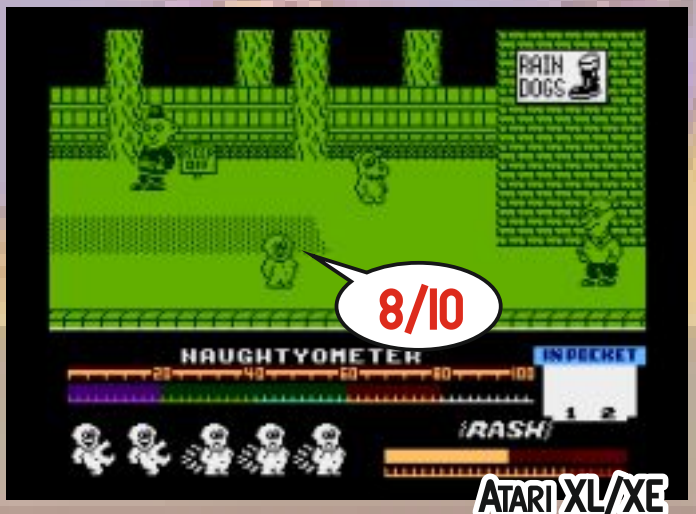
You can carry two items at a time in your pockets. These can be picked up or dropped by pressing keys 1 or 2. Entering rooms to the left or right is as simple as walking through the door, but for some reason, you have to press Enter to go through a door at the top of the screen, which is a bit weird and perhaps unnecessary. You can cause mischief by firing your pea shooter (once you’ve picked it up - it’s up on the shelf on the far right of the first screen) at some of the marauding characters who can be temporarily dispatched but they will always re-appear if you leave the room and subsequently re-enter it. Little Jack can also attain prodigious leaps of which Frogger himself would be proud.

There are 18 objects to collect throughout the game and each can be used to solve a particular puzzle or to be more precise, cause extreme mayhem. The puzzles aren’t too taxing - for instance, if you pick up the glue (which is in the Launderette) and head for Gummo’s Chomping Molars shop, you can use the glue to stick the false teeth together. Or use the horn to terrify the sleeping cat (who is in the Police Station), which will then jump onto the ceiling! Other useful items include fertiliser, soap powder, a battery, a floppy disc, a credit card, a key, a potty and for the more destructive toddlers among you, a bomb! So, what are you waiting for - get naughty!

ZX Spectrum

Commodore 64

Amstrad CPC

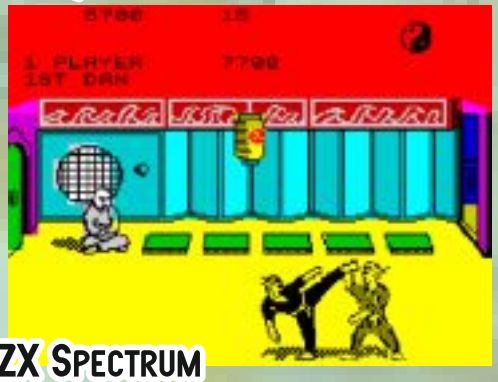


Atari XL/XE

The Way of the Exploding Fist

COMMODORE 64

BBC / ELECTRON



ZX SPECTRUM

AMSTRAD CPC

COMMODORE 16

ATARI XL/XE

When I found out that The Way of the Exploding Fist had finally become available for the 8-bit Atari, I was initially quite excited as I had fond memories of playing the game on the ZX Spectrum back in the 80s. The original actually came out for the Commodore 64 in 1985 and ended up winning Game of the Year at the Golden Joystick awards and was subsequently converted to the ZX Spectrum, the Amstrad CPC, the BBC Micro and the Commodore 16, on which this version is based, which means that unfortunately, it's a 16k game.

On the plus side, the music on the demo mode is converted from the Commodore 64 version and is quite superb. The jingles and sound effects are also very good - unlike the excruciating sound effects used in the C64 original which were lauded at the time. However, being converted from the C16 means that the graphics are pretty basic - if you're expecting anything similar to the wonderful animated sprites (or indeed, the bone-crunching sound effects) of International Karate, don't!

The fighter graphics are pretty small and have a limited range of moves - the acrobatic jumps of the other versions are not supported here, presumably due to memory constraints, and the flying kicks are also conspicuous by their absence. These fighters couldn't jump one inch off the ground if their lives depended on it. You can however, control the speed of the game by pressing keys 1-4, which range from the ponderously slow (4) to almost Bruce Lee like speed (1). The C16 game used character graphics for the fighters, and being a direct port from that machine, there are no Atari hardware sprites used for the combatants, which is a shame. Although small, they are rendered in high resolution single-colour mode, although player 2 is heavily dithered to differentiate him from player 1, which is a little unfortunate, but unavoidable.

Pressing the Start key begins a game against computer opposition. After defeating your first opponent, who is a novice, and frankly, pretty rubbish at fighting, you advance through the "Dan" ranks, with two fights against each rank. I managed to reach the Fourth Dan but the background graphics, which look okay, if a little cartoony (definitely not in the same class as International Karate or indeed the Commodore 64 version of this game) didn't change so I assume the same background graphic is retained for every level. I could be wrong, but considering the game is under 16k in length, I doubt it.

The main plus point for this game is when you press the Select key to begin a two-player matchup. This type of game is always more enjoyable with two human players. In this mode the game is actually playable enough, but it does become a bit tedious due to the lack of available moves, which makes each round resemble waddling back and forth while administering the occasional punch, kick or leg sweep to relieve the monotony.

It's nice to see another old classic game converted to the Atari, but in this case, it's probably a port of the worst version available, with the saving grace of some excellent music, which once again, comes courtesy of the extremely prolific Miker. The game isn't too bad, but with the completely superb International Karate already available for the Atari, it's strictly no contest, if not a mismatch.

SECRETUM LABYRINTH KINGS GOLD

Secretum Labyrinth: Kings Gold is a freeware title from V61 (Video 61 and Atari Sales) programmed by Peter J. Meyer and produced by Lance Ringquist. It requires a 64K XL/XE or compatible computer and is free to download as an .xex file.

There is confusion with some, whether this is a demo or a full-fledged game. While loading, the game calls itself a beta / demo. On the title screen the game calls itself a freeware demonstration. It also refers to the purchasable cartridge game as the sequel. After playing many hours, I personally can attest that this is a full game in it's own right. It may be much smaller in scope to the sequel, but this is a full quest with an ending, as large or larger than any free home-brew titles out there. So... I asked Lance himself if he considered it a beta, demo or full-fledged game and prequel to Secretum Labyrinth: The Legend, and he answered YES. Well, that's that cleared up then!

Th e st or y go es ; m an y ce nt uri es ag o a Ki ng liv ed w ho ha d six tre asures that were stashed away in a hidden labyrinth to which you found the hidden entrance. The labyrinth is guarded by monsters and magical creatures. You must traverse the labyrinth and retrieve the lost treasures. There are scrolls you encounter along the way that reveal more of the story and give hints (and advertising). The backgrounds and walls of the labyrinth are simple patterns, and though the entire map is indeed a labyrinth, the individual screens are generally open rooms and passages of varying shapes and sizes. There are also simple background or floor patterns, made up of pixels against the black background that can be turned on or off, along with the music, at the title / options screen.

The "new" graphics come in where the player / missile graphics are concerned. A complex multiplexing sprite engine, with up to a dozen sprites on-screen at once, all animated and all multi-coloured. Though there is flickering when multiple sprites are on the same horizontal plane, they move extremely fast and there is absolutely no slow-down or stuttering. The sprites are detailed and smoothly animated. I actually like the flickering that occurs when your character is swinging his sword, it makes it seem like a flickering magical energy sword. All in all, it's a very impressive sprite engine, and I must note that I played on a PAL system, so NTSC may well produce less flickering. I was stunned when I first saw so many highly animated sprites in such a rainbow of colours.

The sound effects sound alright, although a bit on the spartan side. The monsters are silent except for their strikes upon you, and other sound effects used are for your weapon, striking a monster and a sound when a monster is killed. Different sounds occur when things are picked up.

The musical score is very "adventure and glory"; an epic-like "symphony" which I really enjoyed. It seems to be just one song, though I have probably only made it half way through the game, but it is flowing and changes, and doesn't get old to me. It fits the game and onscreen action appropriately.

Control is responsive; you move in any of 8 directions with a joystick, and you can hold down the button and continually use or fire your weapon while on the move. There are 5 (6?) different weapons that are chosen with the 1-5(6?) keys. If you use an XEGM, you will need the keyboard. The game play is an action RPG, a bit of a mix of Temple of Apshai, Shamus and Berzerk. You gain experience, rise in level based on experience, and have hit points which, if reduced to zero, results in death. And just like in a standard RPG, hit points also increase with experience when your level increases.

The weapons I found along the way include a sword, hammer, bow and arrows and I believe a mace. I'm not sure what the last one is, but also, on one adventure a 6th spot highlighted for me but showed no weapon and when I chose it, and used it, there is a sound effect, but it didn't seem to do damage to any monsters. The damage inflicted by your weapons increases with your level. You will also find health potions to restore hit points, keys to gain access to certain doors, scrolls containing information, and of course 6 treasures to be found throughout the labyrinth and secret levers that reveal hidden rooms or passages, and sometimes killing all the monsters in a room also reveals hidden doors.

There are many creatures in the game. I'd guess that I encountered 15-20 different creatures and monsters, which vary in hit points and hit damage; some can move through or fly over walls, some can't. Some teleport. All monsters do direct attacks, none have projectile attacks and many of the usual suspects of myth and legend are represented. There are also healing stations that will restore full hit points, and level-up statues that will increase your level and hit-points if you have enough experience points. Potions are used when you grab them, so if you don't need to be healed yet, leave them and come back for them later. If you die, that's it - game over; there is no continue option or save mode.

There are three modes of play in King's Gold; standard quest, alternate quest and random quest, which are chosen by pressing Select at the title screen. Secretum Labyrinth: Kings Gold is a fun adventure that will keep you coming back for more, like the classic dungeon romps of old, with enough arcade-like action and RPG-like adventure to please both the RPG and action / arcade fan. And since it's available as a free download, what do you have to lose?

SECRETUM LABYRINTH THE LEGEND

The Legend is the sequel to King's Gold in the Secretum Labyrinth series, with word of a third installment on the way. Programmed by Peter J. Meyer and produced by Lance Ringquist under the V61 label at AtariSales.com It requires a 64K XL/XE or compatible computer, and a joystick. Users with an XE Games Console will require the separate keyboard.

The story: over a millennia ago lived a Knight named Payton. His Kingdom was invaded and the King had his treasures hidden in a Labyrinth Dungeon. With the help of a sorcerer, monsters and passages were created to protect the King's wealth. I must admit I'm a bit confused if the Knight (Payton) is the King or not, and if not, where this Payton guy comes into the picture or what he has to do with anything. Maybe these things will come to light through the many scrolls hidden throughout the labyrinth. I'll leave that for you to discover for yourself. Your character was following the legends when he bought a box with two items inside that looked like they fit together. When he connected the parts together, he was transported to the Labyrinth Dungeon. There are 16 treasures for you to find to complete your quest. May the spirit of Sir Payton shine on you!

Graphically, The Legend is very similar to King's Gold. Older style simple background graphics, open rooms of various sizes and shapes, with simple patterns on walls - sort of like Shamus, but it uses sprite multiplexing with use of Display List Interrupts for very colourful monsters which are also well animated and move very fast. The sprites are re-drawn, recoloured and animated with better results than in King's Gold. There is still a lot of flickering when multiple sprites are on the same horizontal plane, but the action never slows down - a tour-de-force of software multiplexing on the humble 8-bit Atari. I'm stunned at the fast animated action and rainbow of colours of the many onscreen monsters.

Audio is mono only, including sound effects and music, which changes from area to area in the labyrinth to go with the action, which is a big change from the single song (though with changing parts) of the prequel, King's Gold. However, I didn't hear a single jingle in The Legend that I thought came close to the quality of the multi-part symphony in King's Gold. King's Gold had an epic symphonic sounding song which seems to have been thrown out for mood-setting jingles. You decide which you prefer, but I preferred the King's Gold music.

Sound effects are smart and effective, yet still a bit spartan, like King's Gold. It would have been nice if

some of the monsters made noises. The sounds include weapon strikes upon monsters or yourself and little sort of explosions / implosions when monsters die - and simple sounds for picking up items and whatnot.

Game play is an action RPG. It's very arcade oriented, more like Venture on steroids than Temple of Apshai with the action, but with some RPG qualities in the stats, with hit-points, experience and character level, and your weapons also grow more powerful and faster as you gain levels. Other RPG qualities are secret passages to find, riddles to solve (through the scrolls), and treasure and keys to find.

You also get restart points along the way, so you don't have to start all the way back at the beginning of the labyrinth (like in King's Gold) if you have encountered such an area. Another difference between The Legend and King's Gold is that in The Legend, you don't automatically use health potions when you pick them up, but can hold up to three of them in your inventory and use them at will by pressing the “P” key.

To gain levels you must encounter wisdom idols. If you have enough experience, you gain however many levels and added hit-points as you have experience to achieve. There are also healing idols in the labyrinth. Progress requires you to find different coloured keys for different doors. Quest completion requires all 16 treasures to be found. Scrolls will give you hints in your quest, continue the story, and also provide riddles that must be solved to progress.

There are 36 different monsters to encounter in The Legend, ranging in hit-points and damage they inflict. Some can pass through or over walls, some cannot, and again, some can teleport. All the classics are here from skeletons and ghosts, to griffons and dragons and all the usual suspects of legend, lore and horror are represented. You can acquire up to 6 weapons including a sabre, hammer, bow and arrow, knives, magic wand and a water bucket. These are accessible once you acquire them with the 1-6 keys.

As with King's Gold, there are three modes of play; standard quest, alternate quest, and random quest. Accessible at the title screen with the SELECT key. So it's got loads of replay-ability for many hours and years to come. Again, as in King’s Gold, you can turn the in-game music on or off at the title screen with the OPTION key. Overall, Secretum Labyrinth: The Legend takes what started with Kings Gold to the next level in just about every way - it is an addictive arcade action RPG not quite like anything you've played on the Atari 8-bit before. It loses a couple of points for the simple backgrounds, but otherwise is a pretty top-notch game.

AMOKBOTS

Amokbots is an original cartridge manufactured and sold exclusively by Video 61 under the “V61” publishing label. Programmed and designed by Peter J. Meyer and produced by Lance Ringquist, it is compatible with all Atari 8-bit computers with at least 48k of memory.

The story is what you expect - the same story told for generations now; the robots were made to service man and over time their artificial intelligence “evolves” and they either turn around and enslave mankind or just want to eradicate them! This is the enslave man version, due to a virus introduced to robot troops in the midst of war. You are part of the human resistance, which for all others beside you, Pete Argon, seems to mean a resistance to actually fight for their freedom, so they send you, Pete Argon, in - with one gun. No, it's not a special or powerful gun, it’s really just a pea-shooter. Thank you very much, rest of mankind!

Graphically Amokbots hails back to the golden age of the arcade, as in such titles as Berzerk and Frenzy. This is on purpose in homage to those games, and was originally planned to be a spiritual sequel to those classics, but grew to be more than that. However, it uses an advanced sprite multiplexor, so there are far more enemies on-screen than the 8-bit computer version of Berzerk, and possibly even more than the original arcade game.

Also, Amok Bots is more colourful than Berzerk, as Frenzy was. The average seems to be, including your character, over a dozen animated sprites, and more than a dozen colours. The background / wall colours change from screen to screen, and are slightly more detailed than Berzerk or Frenzy. You get that classic “electrified” effect if your character is killed.

Sprites are mono-coloured but they do have detailed animation. The down side is that there is flickering of the sprites when too many are on the same horizontal plane, but that's expected and well worth all the extra sprites, in my opinion. I'm using a PAL machine though, so there is probably less flicker on NTSC machines. There is text based stat line at the top of the screen and the rest of the screen is all game window.

Audio is sound effects only, which includes laser-blaster sound, similar to Berzerk, from you and the bots. There is also the electrified death sound and an electrified sound for force-field walls but that's it for sound effects. Unfortunately we don't get any classic computer voice synthesis. There is no music, but that is also in line with the classic Berzerk and Frenzy.

Movement is in 8 directions with the joystick, as is shooting, but you must hold the button and press the joystick in the direction you want to shoot. Control is responsive and the Space bar pauses the action. Select returns you to the title screen but also chooses the game mode while on the title screen. Game play is and isn't what you would expect: there are four modes to choose from; easy, normal, challenger and arcade. Arcade mode, the mode I find most fun, is right on the money with Berzerk and Frenzy, except for the lack of the bouncing evil “emoticon” that would come after you if you stayed in one room too long. Rooms are random in arcade mode and the action gets frantic quickly as you go up a level of difficulty with each room.

The other modes are more adventure based. The different rooms stay the same and if you back-track, you go back into the rooms you already completed or left. If you die, you are placed back in the first room and you have to go back through rooms you've already completed to make it back to where you died, where you essentially left off. This is a very annoying waste of time, IMHO, and it should just start you where you left off.

I didn't keep count, but it seems there are about a dozen rooms to complete, at least on the first level,

and you reach a room with a “boss” bot that can take more damage before you kill him. Even if there are other bots left, the level is completed when you destroy him and you move on to the next level. Each game level gets a little bit harder, but unlike the arcade games, each level has many rooms and a boss bot to kill.

In the game you can only fire one shot of your gun at a time and until it hits something, you can't shoot again. The blast will deflect off some walls and there can be two deflections in total before the shot stops. Walls that your shot deflects off could be separated by the full length of the screen and the shot will go back and forth across the room and you are helpless but to run until the shot is finished and you can shoot again. Luckily, only one bot can shoot at a time, and their shot has to finish too, before that bot or another can shoot again. Bots start out slow and lethargic, but in-between shot rates and movement increase with level. Maybe they can shoot more than a shot at a time in the higher levels than I've gotten to, though?

You do not get electrified for touching walls in Amokbots, as in Berzerk, but there are force-fields that go on and off and these will electrocute you. Some walls can be shot through, like in Frenzy. Some bots can move through walls and other cannot. Like your player, the bots can move and shoot in 8 directions too.

You start the game with 7 lives, extra lives being awarded at 2000 points and then every 5000 points thereafter. You lose a life by touching a bot, getting shot by a bot, or touching a force-field. There are 8 different types of bots you will encounter (not including the boss bots) in the non-arcade modes of the game.

The bots are as follows: Omega78 orange Sentry Droid (75 points), Beta34 blue Walking Robot (100), Rho47 white Roving robot (125), Psi24 purple Hovering Bot (150), Gamma52 green Insect Droid (175), Sigma14 cyan Two-eyed Unit (200), Kappa001 yellow Cyborg Head (250), Omega101 purple Master Robot, and other Bosses?

Overall, Amok Bots is an excellent tribute and spiritual successor to Berzerk and Frenzy in arcade mode, and a bit more of an adventure-shooter with multi-room levels and bosses in the other modes. Either way, there's some classic style gaming goodness to be had, with the game style harking back to a simpler age and it pulls it off pretty well. If you like classic arcade shooters like Berzerk, you'll like Amok Bots.

Retronics Collector's Edition Set 1 by David Sherwin

Several months ago, I saw a post on the AtariAge forums from a member who was offering a set of games for sale. They were Polish games, Jurek Dudek explained, and part of a series that was celebrating the best of them from the 1990s. I'd sort of heard of some of the software houses, like L.K. Avalon, as I'd acquired a few disks from other collectors over the years, but hadn't been really impressed with them, packaged as they were in cheap plastic outer cases that were slowly dissolving the floppies inside. I hadn't heard of ASF or Mirage, two other Polish software houses that produced, as it turns out, some pretty amazing games during that era. I decided to go for the entire first lot of Retronics' "Collector's Series" and got my games about a month later. And boy, were they worth the wait!

What makes these games so fascinating to me, a Canadian A8 owner and user since 1983, is that the "Collector's Edition" represents a missing generation of Atari games that North Americans didn't get to experience during the commercial lifetime of the Atari 8-bit computers. There are virtually no NTSC platformers, point-and-click graphic adventures, and advanced side-scrolling shooters because the North American market died around 1988, just as these gaming genres became popular on other competitive systems. In Poland, though, it's apparent that game development for the A8 platform proceeded apace with game development for more advanced 16-bit systems, and some of the graphical styles and music popular in 16-bit games were reproduced here to jaw-dropping effect. Jurek says that the Atari 8-bit was the most popular computer platform in Poland in the early '90s, and so the A8 got the best coders and games for the 8-bits. Some of the more successful titles were eventually ported to other systems, like the C64 and the Amiga, but the Atari 8-bit was usually first. That in itself must have been unique in the home computer wars!

Jurek Dudek has treated each of the ten games in the "Collector's Edition" with great care and attention. Packaging is outstanding and the material therein is, in some cases, pretty much essential to successful gameplay, even if the text is in Polish. Each game is contained in a sturdy outer corrugated box; inside players will find a detailed game map, a "thank you" to Atari fans (in Polish), and a quick reference card. The inner box is then contained in an outer box and a dust sleeve. It's pretty similar what you would've found in a release from Sierra On-Line or Infocom in the late '80s and definitely gives these games the feel of a high-end purchase. Believe me, if you acquired any of L.K. Avalon's disk releases that were packaged in flimsy plastic sleeves with simple squares of instructions, you'll find the "Collector's Edition" stunning.

Sadly, the "Collector's Edition" has not been a strong seller and future plans are now in doubt. Retronics has said that they had planned to release a total of 100 games in the series, and that they represent the best of Polish gaming. I can only hope that we give them enough incentive to allow them to carry through. Now on to the "Collector Series" games…

1. Klatwa – "The Curse"

In a nutshell: It's like a Sierra On-Line Graphic Adventure for the 8-bits! Well, sort of. When the 8-bit game market in North America began to collapse in the mid-'80s, nothing made me more envious of the emerging 16-bit systems than the new, exciting graphic adventures that were released to take advantage of these machines' advanced graphic and sound capabilities. I mean, I almost caved in and bought an ST with my allowance when I saw screenshots from King's Quest, but mightily resisted all dark urges that came my way. Well that's great, I'd think, but I'm not giving up my Atari 800. Back to Joust for me! Surely the A8, with its late '70s computer architecture, isn't remotely powerful enough for one of these adventures, right? Wrong!

Klatwa ("The Curse", 1991 L.K. Avalon) ably demonstrates exactly what can be achieved when coders either don't care about a computer's limitations and are skilled enough to overcome them, or don't have any other choice. Or perhaps both issues were a factor for L.K. Avalon in the early '90's: Jurek Dudek tells me that the Atari 8-bit, with its relatively limited specs, was the

most popular computer platform at the time. It's probable, though, that modern gamers would have approached its games for the A8 expecting the sophisticated graphics and music that were prevalent on competing (and more technologically advanced) systems. To my surprise, Klatwa (largely) pulls off this magic trick.

Ok, here's my disclaimer for this game: Klatwa was cracked ages ago, and there are at least four versions of it floating around the 'net. Some very nice person (or group of people) hacked the original PAL game to work with NTSC machines, and it's this version - and not the Retronics disk game that came in the package - that I played. I couldn't get my Retronics game to run, and it seems reasonable to assume that some NTSC gamers may experience difficulty playing this version of the game. There's also a version of the game with hacked, translated English text, but I wasn't able to find a copy of it for testing.

Klatwa is a typical point-and-click graphic 'quest' game - there's not much unique here - but it's still pretty neat to see it playing on our lowly A8s. The screen is divided into two portions: the upper portion is the playing field, while the lower part of the screen contains the text box from which gamers select game options. Gamers use a cursor (in the form of an arrow) to either move the player around the screen or to access the option text-box. Controls are similar to that in Magia: you select objects of interest on the screen (such as books or treasure chests), and then access the text box to select an option to let you examine or use that object. It's not the fastest system, and led me to vague feelings of rage and frustration, but it does the job in its own arthritic, unlovely way. Sadly, I could not get Klatwa to work with a mouse, which would have been optimal for this game.

Klatwa's game map is included as a poster in Retronics' release, and it's a necessity here, as the castle is quite large and it's easy to get lost without mapping out the playing area. The original L.K. Avalon disk version of Klatwa came packaged in a flimsy plastic sleeve with no map and minimal instructions, and I cannot tell you how wonderful it was to play the game again with a complete map in a professional package. It's almost like it's a different (better) game.

Graphics are monochromatic but impressive. Players don't have free movement around the room - pick a point of interest, and the game automatically moves the player to that point - but animations are smooth and the pseudo 3D perspective is done really, really well here. The in-game music is simple, but effective, and is largely unobtrusive during gameplay.

Klatwa is an amazing feat of programming and a pretty fun game, too. While not the most entertaining of Retronics' "Collector's Edition" series of games - that award must be given to the amazing Hans Kloss or perhaps Miecze Valdgira, but it remains a singular example of an effective graphic point-and-click adventure for the Atari 8-bits.

Graphics: 9/10 **Music/Sound: 7.5/10**
NTSC compatible? Problems may or will be encountered; hacked versions available.
Gameplay 9/10 **Overall 9/10**

2. Miecze Valdgira
In a nutshell: Shadow of the Beast meets Castlevania. Awesome! Miecze Valdgira (ASF) is Jurek Dudek's favourite game in the first ten titles of the "Collector's Edition" re-releases. Of course it is! With its combination of fast action, problem solving, and an intriguing storyline, this is one of the highlights of Polish gaming of the '90s and, really, of the entire system itself. Your character is an adept of magic who has decided to explore a castle that was cursed by a powerful sorcerer before his death. While I'm a little sketchy on the rest of the details (thank you, Wikipedia!) you have to recover five pieces of a pentagram in order to break the curse.

You play an ugly little troll or demon (transformed from human by the sorcerer?) who, naturally enough in a cursed castle, encounters many hazards and enemies throughout his quest to retrieve the five pieces of the pentagram. Fortunately, your character isn't helpless, as he's able to spit flames from his mouth in order to roast his enemies. Fire breath isn't always enough to defeat enemies, so a combination of stealth, timing, and firing may be required to advance through screens. If that isn't

enough, there are a number of 'dead end' rooms which conceal secret passages that can be revealed by blowing them up with explosives (found at a number of points through the game). The included high-quality map is of great assistance in planning strategy without having to map out the entire castle oneself.

Graphics are amazing and amongst the best seen in a scrolling platformer. The player is large and detailed and the animation is nice and smooth. Enemies fly in detailed formations, and each type of enemy has a distinctive flight / hover pattern that's both a treat for the eyes and a pain for the wrists. Typical ASF graphical trademarks, such as elves and dragons, are scattered throughout the castle. I've seen video longplays of ports of this game to 16-bit systems, and while there's an obvious upgrade to the graphics and music, and it's not as striking as you might think, which just goes to show you what a stellar job the coders did for the original A8 game.

Miecze Valdgira spawned a sequel (which is, if anything, even more impressive graphically) that is planned for a later release… if the "Collector's Editions" continue. Let's hope it does - I was simply amazed with this one.

Graphics: 9/10
Controls: 9/10
NTSC compatible? Yes
Music/Sound 8.5/10
Gameplay 9.5/10
Overall: 10/10

3. Operation Blood
In a nutshell: It's a clone of Taito's 1987 arcade smash hit Operation Wolf. Polish software house Mirage (1992) flipped the Commando concept on its side and turned it into a side-scrolling shooter instead of an overhead blaster. It's still just as much a - ahem - blast, and new challenges are created by the scrolling action and the inclusion of different types of enemies and / or covers. Some of the levels scroll from the left and some scroll from the right - and this means that you can't necessarily employ the same strategies for all of the levels. Gamers have a perspective on the field that looks down the sight of a (machine gun), and enemy combatants pop up (and try to kill you!) as you scroll across each level. In the meantime, armoured vehicles and helicopters will roll or fly around, and generally try to kill you. Fun!

Interspersed with enemy soldiers are a number of non-combatants such as civilians and EMS (here represented by old-timey looking nurses carrying a stretcher), which are all amusingly animated and whom you obviously shouldn't target. I couldn't help myself from popping off a few shots at them on occasion, though. Mwah ha ha!

I'd also like to extend a 25 year-old thanks to Mirage for, finally, including bilingual instructions on the title screen. I know you don't really need them in a shooter, but it lets Anglo players get right to the heart of the action, and properly acknowledge the people who worked on this game, too. Also nice and of note: Operation Blood comes on a red floppy, a change from Retronics' black standard disk. Ammo and health are, needless to say, in limited supply, and while you can pick up more on the battlefield, there's some strategy involved in the game. Do you shoot the tanks and helicopters first, or get to that ammo pack? It's decisions like these that make or break any blaster, and it all works here.

The graphics are interesting and a joy to behold. Enemies are scaled, which just isn't something that I'm used to in an Atari 8-bit game. Large and detailed enemy soldiers appear in the foreground (should I admit that I like to shoot them in the crotch?), and much smaller ones appear in the background near buildings. It lends the game a very impressive false sense of perspective that really does give the impression that the playing field is much deeper than it actually is. There's also a surprisingly wide variety of enemy combatants, such as the "large" soldiers down to the "rollers" who, well, roll across the screen only to pop up and start shooting at unexpected and inopportune times. The graphics capabilities shown on the computer seem to be much, much more demanding than they are in Commando, which makes the fact that it works this well all the more impressive.

When I first played Operation Blood I thought "this would make a hell of a lightgun game!". And then I thought of my horribly inaccurate XG-1 (last used for Crime Busters), and realized that this would have been a terrible, terrible idea. I have since discovered that a Light Gun version of Operation Blood was actually released,

but I have never played it, so I am not really in a position to say whether it was an improvement. Joystick control is pretty precise, although my seven year-old found that he couldn't centre the joystick (and firing sights). Yes, yes, I know this isn't a game for children, but I couldn't find a working copy of Mr. Do! for him, okay? The map included with the instructions isn't terribly necessary for this SSS, but it's still nice to have and acts as a preview or incentive for weaker players (hand raised here) who may find themselves to be challenged by the game's difficulty.

My only gripe with Operation Blood is that the game re-loads, by disk, every time you die. With the advent of flash drives and SIDE carts, I'd forgotten what it was like in the days when it was just me, my 800, and my 1050 drive… and those long load times aren't a bonus. Still, I had to grit my teeth and suck it up, as I haven't found a .XEX file for this one, and the game is just that good that it's worth all the waiting.

Graphics: 9/10
Controls: 9/10
NTSC compatible?: Yes
Music/Sound: 9/10
Gameplay: 9.5/10
Overall: 9.5/10

4. Hans Kloss
In a nutshell: It's Impossible Mission. With Nazis! Finally - I've discovered that someone actually coded Impossible Mission for the Atari 8-bit family! And turned it into a World War II themed treasure hunt! Actually, Hans Kloss is even cooler than that: it's a game based upon a fictitious Polish hero who was featured in the popular serial Stawka większa niż życie (please don't ask me to pronounce that) and in multiple other dramatizations, including a movie that was released commercially in 2012. Too bad that this one hasn't been given English subtitles. Damn you, YouTube!

I didn't have much luck with the handsome (Polish) instructions, but this is essentially, you know, Impossible Mission, and controls aren't too difficult to decipher - even for non-Polish speakers (all in-game text is in Polish). The game includes a beautiful full-colour map of the stronghold which makes strategising and gameplay much easier (and not to mention more fun). As Hans, the player must navigate his way through a Nazi stronghold in order to steal top-secret plans and, of course, save the world. Wikipedia is my friend! The plans have been dissembled into their component parts and hidden throughout the stronghold: Hans must traverse roving 'floor mines', locked doors, physical barriers, and elevators / lifts in order to collect all the missing pieces and reassemble them into a working picture. You'll get a glimpse of the entire plan every time you find a piece of it.

Naturally, Hans can't rely on his amazing good looks and charm to get past the numerous gaming hazards, and gamers who forgo the included map will have to map out screens if they wish to advance through it. This isn't an arduous task, but it also doesn't suit impatient, trigger-happy gamers who just want a splatterfest. Despite its looks and obvious influence, this isn't Wolfenstein, people! Planning and strategy - not weapons skills - are the virtues that are rewarded here. Hans may be tough, but he leaks "health" (composed of two separate elements) like a sieve, and unnecessary and wasteful moves will quickly kill him. It's annoying as hell, but quickly gets across the point that strategy is supreme in Hans Kloss.

Controls are something of a letdown. Platformers aren't that much fun if jumping and moving around is too difficult, and too often Hans died a nasty death when I couldn't move him up to the next platform. Precision jumping often works only rarely and only when time isn't an issue, and it quickly devolves into an exercise in frustration when you're trying to beat the clock. You know what I'm talking about here, folks. Joystick response also tends to be sluggish and unresponsive - or at least it is on my machine. Hans is slow, and that, more than anything else, made me want to quit the game in frustration. I'm not sure if this is a NTSC translation glitch, but it's quite aggravating.

The graphics really are top-notch here and contain many amusing touches. Hans does, for instance, recharge by picking up "good" health powerups (coffee and (turkey?) drumsticks), while "poisoned" food results in an animated Hans very visibly throwing it back up. 8-bit animation at its finest! The stronghold

is also decorated with miniature Nazi iconography and portraitures, which does lend the game its sense of place and purpose, much as it does in the Wolfenstein games. Hans is recognizable as a uniformed soldier, and gaming graphics are crisp and detailed. There's even a nicely-drawn title screen illustrating a (smoking) Kloss, and you'll get another if the game is successfully completed. Kloss didn't like my modded NTSC 600XL very much, though, and I saw the occasional graphical glitch pop up in gameplay.

It's clear that the coders here wanted to develop a game with modern (circa 1992) gameplay, graphics, and pretty decent in-game music, and it's remarkable to the extent that they succeeded with computer architecture from 1979. If the game had been released in NTSC markets, and slightly graphically retooled as an Indiana Jones or Wolfenstein branded adventure, this would have been a sure-fire international hit. But Hans Kloss is a winner by any standard and deserves repeated gameplay.

TIP: Can't seem to win? The Secret Plans have been scattered into twenty separate pieces - eleven form the text and nine comprise the picture - and you'll need to find all of them in order to complete your mission. Careful use of food items - don't gorge! - will ensure that you have enough energy to last the entire mission. Take all keys that you find, and remember that some rooms can only accessed by falling down four or five levels (you won't be hurt) or by going through locked doors that only seem to lead to dead ends.

Graphics 9/10 **Music/Sound 8/10**
Controls 7/10 **Gameplay 9.5/10**
NTSC compatible? Some graphic glitches
Overall 10/10

5. Dwie Wieze
In a nutshell: What the hell? I really, really wanted to love this game. Dwie Wieze or "Two Towers" (ASF, 1993) has to have the best graphics I've ever seen on any 8-bit platformer not specifically programmed for the VBXE. The graphics are that good. It's such a pity that the game is terribly let down by poor controls and mediocre gameplay.

All instructions are in Polish, as is standard for these releases, but the gaming environment and standard controls will be familiar enough to allow most gamers to start playing without too much trouble. You are, apparently, the Sorceress Isis, and must traverse four different areas (collecting various items along the way) in order to complete your quest. Isis is the most detailed character I've seen in an Atari 8-bit game, and she's large, distinct, and beautifully animated. Her hair actually moves when she jumps - amazing! Most of the game's graphics are equally impressive, although it's likely that most gamers will be spending their time trying to avoid being killed by all the enemies to be impressed by how pretty they all look. When Isis dies (and she will, frequently), though, players are treated to a very nice animation where she does a double face-palm and disappears. Weird, but pretty - just like the rest of the game.

Oh yeah, that - Dwie Wieze is almost impossibly, ridiculously hard, and I was only able to make some headway in the game by playing one of the cracked versions that offers unlimited lives. Innumerable obstacles can be navigated only with precise timing and exquisite control (and not to mention the need for a top-notch joystick), and the game quickly deteriorates into an arduous slog. Is this an NTSC thing? Am I just a terrible gamer? I'd doubt my own abilities, but a view of some of the gameplay videos that have been posted on YouTube show that mastery of this game seems to elude most players.

Dwie Wieze isn't a broken game - far from it - but it only rewards the most patient, exacting gamer. The rest of us can only look at all of the pretty pictures and dream what we might see next - if only we had the patience, or skill - to get there.

Graphics: 10/10 **Music/Sound 8/10**
Controls 6/10 **Gameplay 6/10**
NTSC compatible? Yes **Overall 6/10**

6. Caveman
In a nutshell: A Paleolithic Platformer. Caveman (Mirage, 1993) is a simple platformer where gameplay is reduced

to a survivalist treasure hunt. You can't - as far as I can tell - shoot or kill any of the game's innumerable enemies, and most of the game's physical obstacles will harm the player's health or kill him on contact. For a hulking brute, this Caveman is pretty fragile. So much for the touted health benefits of the "caveman" diet!

Caveman is a standard platformer. What's not to like in that, right? The answer is, unfortunately, "quite a lot". The problem isn't precisely the graphics: Polish games from the '90s - as we've seen with the rest of this collection - look pretty spiffy, and our hero is suitably well-muscled (not to mention naked). The backgrounds are fine - plants and animals are at least visible - but certain hazards are hard to discern and even harder to avoid. There are these… things (I can't be more specific than that) on the ground on many of the levels, and they're incredibly hard to spot because they tend to blend into the background. I would be more impressed if they appeared to be part of the game's overall difficulty, but their distribution seems to be chancy and they're haphazardly placed through the game. It makes the game harder than it should be, and at the same time makes it less fun. It's quite annoying.

Caveman alternates between day and night screens (at later stages in the game), and this provides some graphic variety even if gameplay doesn't change that much. Many of the screens also contain multiple levels, and ladders to reach them, although it's not clear that there's anything of value in or on the upper levels of most of them. Caveman also has to enter caves in order to pick up items to complete his quest, but they all have many of the same hazards and play much the same. The caves, with their boring timed jumping sequences, were by far my least favourite part of a game that doesn't come with many highlights in the first place.

Graphics: 6.5/10 **Music/Sound: 6/10**
Controls: 5/10 **Gameplay: 6/10**
NTSC compatible? Some graphic glitches
Overall: 6/10

7. Fred

In a nutshell: It may look like a cartoon, but he's not a Flintstone. Fred (L.K. Avalon, 1991) is the platformer that Caveman wishes it was. While its concept is somewhat bizarre - and controls somewhat counterintuitive - this is a far superior effort.

Fred, like Caveman, is a side scrolling platform with nice graphics. Except that you're not naked in this game! After a fairly nice (if bland) opening screen, we're off and - what the heck? It looks like Fred's got himself stuck inside of some bizarre cactus land filled with green menaces that look an awful lot like frogs. And since I can't read Polish, and therefore the nice included instructions, I'm sticking with frogs. Fortunately Fred, unlike the titular Caveman, can fight back, and has a limited supply of rocks and, later, something that looks a lot like a water sprayer (or an air horn) with which he can kill the "frogs". I'm not sure why water would kill frogs, but work with me, people! Fred can ramp up his offence by finding additional objects along his route, such as more powerful weapons (including dynamite, which kills all enemies on the screen), shields, and extra lives. Each weapon is effective against a certain range of enemies, and it's useful to have an assortment of items as you work your way through each level, although none of them are ultimately as useful as the shields and "water spray". Bonus items are hidden in what look like, um, piles of poo, although I'm sure that it's meant to be something much more elegant. Some items are also hidden in jugs / amphorae, and Fred must hit both the "poo" and the jugs in order to discover what items might be hidden in each container.

Each of the ten levels is fairly varied, and there are a

number of different types of enemies and obstacles scattered throughout the game to keep things interesting, such as the ever-present frogs and birds. Or are they bats? The levels have distinct themes, such as "desert" and "underwater". There's also Level 8, which includes a message (in Polish) to gamers that is cleverly spelled out in the level's rock walls. Instructions include a full-colour map of all ten levels which makes planning your route through the game a breeze.

There's a nice pacing to the physical obstacles in the game as well, with standard precision-timing obstacles as well as spatial puzzles that must be solved by using a combination of timing, climbing, jumping, and firing. In addition, some obstacles require use of specific items rarely found in the game. I won't spoil the surprise beyond stating that you've got to collect and hoard as many shields as you can in order to complete the game. The end of each level requires gamers to collect a set number of "bonus" objects in order to advance to the next stage, and it's a nice level of additional complexity that elevates this platformer beyond a simple run-and-jump exercise.

Graphics are large and cartoony and generally pleasing to the eye, although the palette is kind of sour and dour. I wasn't expecting B.C.'s Quest for Tires here, but some colour variation would have been nice. The in-game music is also effective and remains quite enjoyable even after repeated listens. That's something when you consider that you might be playing Fred for hours on end. It's also something that was missing from most North American releases, and it's interesting to see how much it increases the enjoyment level of general gameplay.

The Atari 8-bit family sure didn't get a lot of platformers during its commercial life - especially in North America - and it's interesting to see that some companies did try to fill this very large hole in its library with interesting and innovative games. While Fred certainly isn't going to make anyone forget Super Mario Land or the bazillions of other platformers that came in its wake, it's a pretty good effort and a great way to spend some time on the Atari 8-bit.

TIP: Can't get through a certain hazard? Use your shields! Fred is impervious to cacti when using his shields, and you may find that you have to walk through cactus plants in order to collect bonus items at the end of each level.

Graphics: 8/10
Controls: 8/10
NTSC compatible? Yes
Music/Sound: 8/10
Gampelay: 8/10
Overall: 8/10

8. Kult

In a nutshell: it's the secret Love Child of Zybex and Scramble! And totally badass! Now this is my type of game: a side-scrolling shooter with some awesome (if monochromatic) graphics and cool music. And throw away those manuals! (or, in this case, just keep them in the box). I suppose that there are a few gamers out there who have grown up solely on games like "Solitaire" and "Shanghai"… but you know you're not one of them. It's time to bring out the big guns, and Kult (another stunner from ASF, 1992) gives you plenty of them and doesn't waste any time with tutorials or playthroughs. The ease of learning was also a nice change of pace, as there's no need to translate the handsome instruction card from Polish.

There's a basic plot here, but all I can tell you is that it involves something about a helicopter. A Super Helicopter! That hovers without user input (the enemies are enough for me)! And pulverizes enemies with a wide assortment of weapons! Awesome. In Kult, like most SSS games, the player controls a vessel that begins from the left-side of the screen and progresses to the right while avoiding physical barriers and enemy targets. The pretty buildings in the lower half of the screen aren't merely

eye-candy in Kult: they're also obstacles that will blow your 'copter apart if you touch them. Palm trees, sand dunes, and gravestone crosses (all rendered clearly and beautifully in hi-res, I might add) are apparently a lot deadlier than they look… and it's easy to be distracted by them, because the graphical environment here, as is typical in an ASF game, is fantastic. At the same time, you'll have to avoid waves of enemies that approach in formation and undulate hypnotically back and forth across the screen. One AtariAge forum member described it as "very Zybex". Which it is… if Zybex were in black and white. Each stage has a different theme, such as the "country and city" theme of Stage 1, or the "Desert" of Stage 2, and the graphics are beautifully drawn. There are nice little graphic details throughout the game, and I found myself constantly distracted (and flying into obstacles!) by all the pretty sights. And there are lots of things going on in the screen, which is divided into the playing area and your helicopter's display. Each level has a boss, which is large, nicely detailed, and suitably difficult to defeat. Okay, it's the same boss at the end of every stage, but it's nice to see this feature included on an A8 game.

One of the nicer touches is the 'radar' at the bottom of the screen, which gives an advance preview of future enemy waves. It's touches like this which really distinguishes Kult from the rest of the pack. Kult also has the one of the best title sequences I've seen in a '90s A8 release, with some great camera angles and a high-score table with a pulsing soundtrack that's very T2K. This is high-quality work all-around.

Kult is another punishingly difficult ASF release, but at least you've got the opportunity to earn plenty of powerups to power through enemy waves, and the game never becomes difficult to the point of frustration. There are several different types of munitions, and, interestingly, they actually do have varying impacts on enemies, so even some element of strategy becomes possible in what should be a typically standard shoot-'em up. Still, Kult isn't easy, and gamers will be kept on their toes (and their fingers on the trigger) as they blast through succeeding waves of enemies. It's an amazing game that will keep arcade gamers satisfied - and returning - for more missions. I'm getting excited just writing about Kult - and thinking of all the areas I've yet to explore - and am definitely making room for this gem in my permanent collection. Kult worked flawlessly on my NTSC machine.

Graphics 9/10
Controls 8/10
NTSC compatible? Yes
Music 8/10
Gameplay 9/10
Overall 9.5/10

9. Magia ("Magic")
In a nutshell: It's Airball! It's Knight Lore! Wait - what the hell is it? Magia (Mirage, 1993) looks like the results of an exotic breeding experiment involving the amazing A8 conversion of Knight Lore (see the review in Excel No. 1) and Airball. It also plays like, well, a point-and-click adventure on the ST from 1986. But in Polish! Wait, What? And more importantly, why?

I'd love to have written a fun, in-depth review of this game, but this is just one of those titles that non Polish-speakers are just going to struggle with: input and interaction with your player is through scrolling menus (in Polish), and, while you can get around a few screens through inspired guesswork and translations, it really isn't ideal. I know that I'm supposed to explore and look for things in order to do something, but damned if I know what that might be. I plugged in my ST mouse, just for fun, but no luck: input is by joystick only.

I get the feeling, though, that Magia wouldn't be that much fun even if I knew some (any) Polish. Turning a dungeon crawler (in false 3D perspective) into a point-and-click adventure is a neat idea on paper, but the execution here is somewhat questionable. It's click, scroll down the menu, pick an option, wait, move a bit, and then repeat… over and over and over again. It takes a

loooong time to move your character (who looks like a cowled mage) anywhere on the screen, and the joystick isn't particularly responsive. Really, removing the menu interface would've accelerated the game and made it much more fun to play. In any case, none of the YouTube walkthroughs that various gamers have posted have shown anything that suggests that you can do a quick run through this one.

The graphics are lovely and there's some cool (if repetitive) in-game music. I just wish that there had been more on offer to induce me to stick around and enjoy it.

Graphics 8/10
Controls 6/10
NTSC compatible? Yes
Music 7/10
Gameplay 6/10
Overall 6/10

10. Władcy ciemności

In a nutshell: No, no, noooo! I wanted to end the "Collector's Edition" on a high note, but must tragically report that I couldn't get this game to play on my NTSC machine. Władcy ciemności ("Lord of Darkness" L.K. Avalon, 1993) is, like Klatwa, a point-and-click graphic adventure that's as handsomely boxed as every other entry in the "Collector's Edition". It's even got a great map of the entire castle, which will be of great assistance to (PAL) gamers who take on this particular challenge. ładcy ciemności has been cited by a number of gamers as one of the greatest releases in the '90s - it was later ported to the C64 - and has gained new fans while retaining its sterling reputation from a generation ago. Sadly, I was unable to obtain a .XEX file that would allow me to play the game in any form. From the longplay videos I've watched, I can attest that its combination of sharp visuals and innovative gameplay make this one a must have for those who can play PAL 8-bit games.

Graphics: 9/10
Controls: N/A
NTSC compatible? No
Music/Sound: N/A
Gameplay: N/A
Overall: N/A

BONUS MUSIC CD

Jurek has included an excellent bonus CD as a surprise gift for purchasers of five or ten games in the first set of the "Collector's Edition" series! The CD contains 20 selected music tracks from all of the first ten games in the series as well as a bonus program track at the end of the disc. All of the selections are excellent, and show how much the A8 platform could deliver, when properly programmed, in the 1990s. It's simply astounding that no software houses in the NTSC market really bothered to explore in-game music at any point in the commercial life of the A8 platform in North America, and a listen to any of these tracks, either on the CD or during gameplay, shows just how much they enrich the basic gaming experience. The tracks from Hans Kloss (track 8), Dwie Wieze (tracks 9 and 10), Fred (track 14), and Kult (track 15) show that the A8 could produce electronic music that was as compelling as anything put out by more advanced computer platforms in the early '90s. While I'm sure that other A8 chiptune collections have been individually collated for private enjoyment, this is the first item that I've seen that has been commercially produced to support an existing software release. The "Collector's Edition" bonus CD fills a void in my A8 experience that I hadn't even known existed until now. **10/10**

Just as this issue was being finalised, Jurek Dudek announced that he will, indeed, move ahead with his plans to release more Atari 8-bit games! Series 11-30 is now available on tape and will feature many more hits, including **Bang! Bank**, a conversion of the arcade classic Bank Panic. A release for floppy disc users is imminent. While Jurek will no longer be issuing these games in deluxe "big box" packages, Atari users will at least be able to continue to enjoy his releases.

Q: Tell us about yourself. Why are you interested in bringing these games to the Atari A8 platform in 2017?
A: My name is Jurek Dudek, I live in Poland. I am big fan of retrocomputers, mainly Atari, Commodore, Sinclair and Amstrad CPC, but I like also other platforms. Some time ago I started collecting games for them. I have noticed that titles released for Atari (it was the most popular platform) in the 90s in Poland have poor quality, and therefore it is very difficult to load them into Atari. My idea was to produce them again, in small quantites - for collectors, but using better tapes. All games, released on disks and tapes are 100% clones of originals, keeping the copy-protection systems. In this case, games are not cracked or modified in any way. I have bought licenses from copyright proprietors.

Q: You've already released titles 1-10 for the platform. What are your future plans for the "Collector's Set"?
A: It is not known yet. Maybe I will continue with it, but everything depends on demand. For now it is a bit too low. I have dozens of titles ready to be released.
Q: Many North American and Western European gamers aren't familiar with the Polish Atari scene. How popular was the system in the '90s, and how active is the scene today?
A: The system was very popular at the end of 80s and in the 90s. The scene is big today - our Atari parties (like Silly Venture) are being visited by hundreds of people. Many new demos and games are still developed.

Q: What was the home computer software scene like in the '90s? I heard of LK Avalon, but none of the other software houses that are featured in the "Collector's Edition" titles.
A: In the 90s, many Polish companies started to release games. There was LK Avalon (probably the biggest one), A.S.F., Mirage, Stanbit, Domain Software, L.K. Fire-Bird, and many more. Business was good for them at that time, even despite pirated games being easily available.

Q: I've seen or heard of many truly amazing Polish turbo cassette players over the years. Are cassettes still popular in central Europe? Is this the reason you've also released the "Collector's Edition" on tape?
A: To be honest, tapes are not very popular now, but many tape lovers still collect them and use them. The main problem is to find a working tape deck, but soon I will release something to help people who do not have an Atari tape deck...
Q: I saw the odd release from LK Avalon sold around North America - "Klatwa" ("The Curse") being one of them, but most of the other titles you've released are unfamiliar to me. What's your favourite game amongst the titles you've released?
A: My favourite games are Miecze Valdgira (Swords of Valdgir), Fred and Hans Kloss.

Q: "Wladcy Ciemnosci" has been cited by more than a few gamers as the best game that they've played. Why is it so highly regarded?
A: Władcy Ciemności (Lords of Darkness) is a sequel to Klatwa. Very similar game.
Q: "Hans Kloss" has a particularly intriguing backstory: from what I understand, this game was developed from a television series portraying a fictitious World War II hero who infiltrates Nazi headquarters to steal their secret plans. Can you tell me anything else about the development of this game?
A: Hans Kloss was indeed a very popular television series. Your role in this game (as a Hans Kloss) is to steal secret plans from Nazis. The playability of this game is really high.

Q: Have you actually watched "Stawka większa niż życie" (the TV series on which the game was based?) Is the video game faithful to it?
A: Of course I have watched it! It is difficult to find Polish people who haven't. The series was extensive but the game is not very faithful to the series!
Q: You've certainly put a ton of effort into these releases - documentation, boxes, and extras are all top-notch. What a labour of love! Why did you do this instead of just releasing the games in cheap plastic clamshells, like the one in which my original version of "Klatwa" was encased?
A: I have also Classic version which is a plastic clamshell, inlay (including manual and description) plus code card if necessary. Collector's Edition is a piece of art, including also a big-size map and better poligraphy.

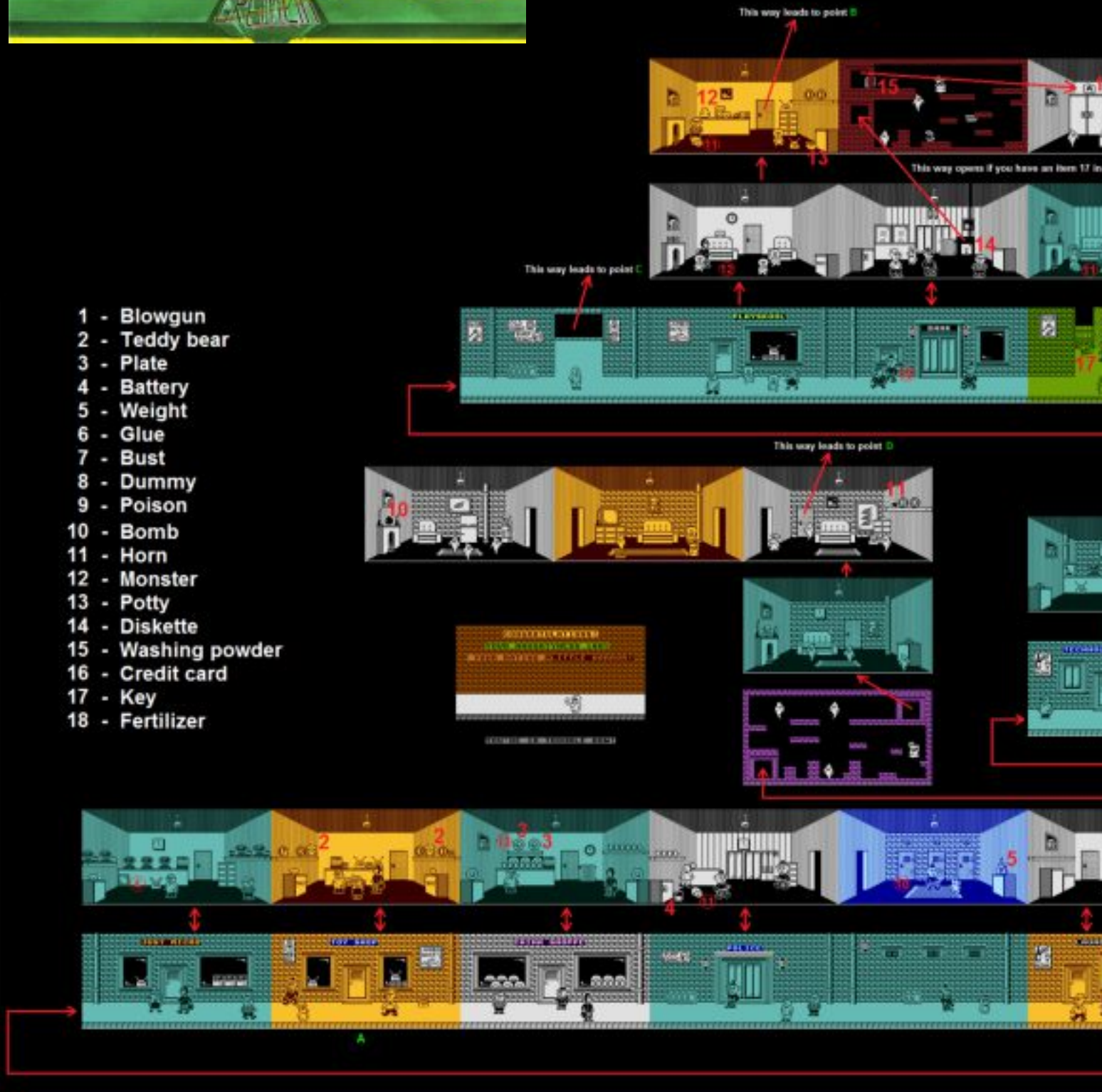
Q: The bonus "Music Collection" CD has some amazing A8 music and a few surprises. What can you tell readers about Track 21?
A: Well, track 21 is a program track for Atari. You can load it to Atari from tape, recording signal from CD to tape first. Or, you can use your PC for ripping it into WAV file and converting into .CAS for using in emulator. This program is a music box for playing in Stereo all tracks included in first ten titles. Vol.2 is also ready, with music from the second set of ten titles.

Q: How can gamers buy the "Collector's Edition" games? How do they contact you?
A: Tape versions are available on ebay, it is also possible to buy all titles (disk, tape, Classic) via e-mail: **zamowienia@retronics.eu** - please keep in mind that the number of every title is limited, therefore in the near future I may run out of stock.

JACK TH

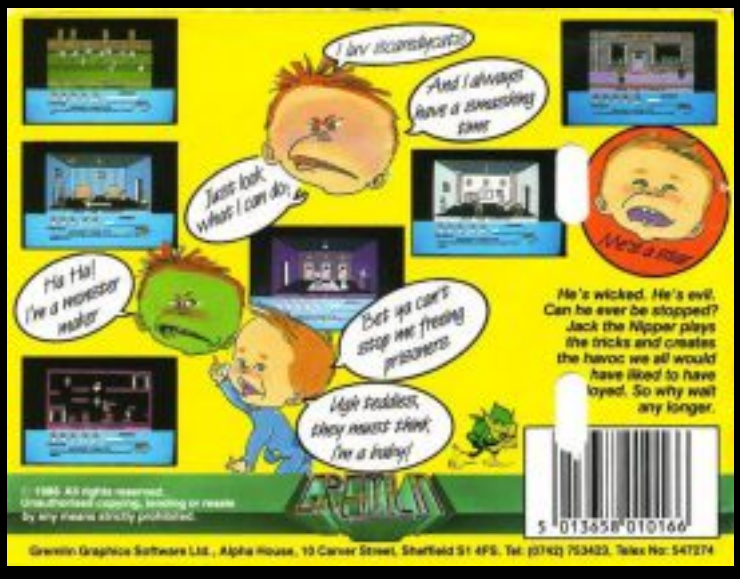
GAME

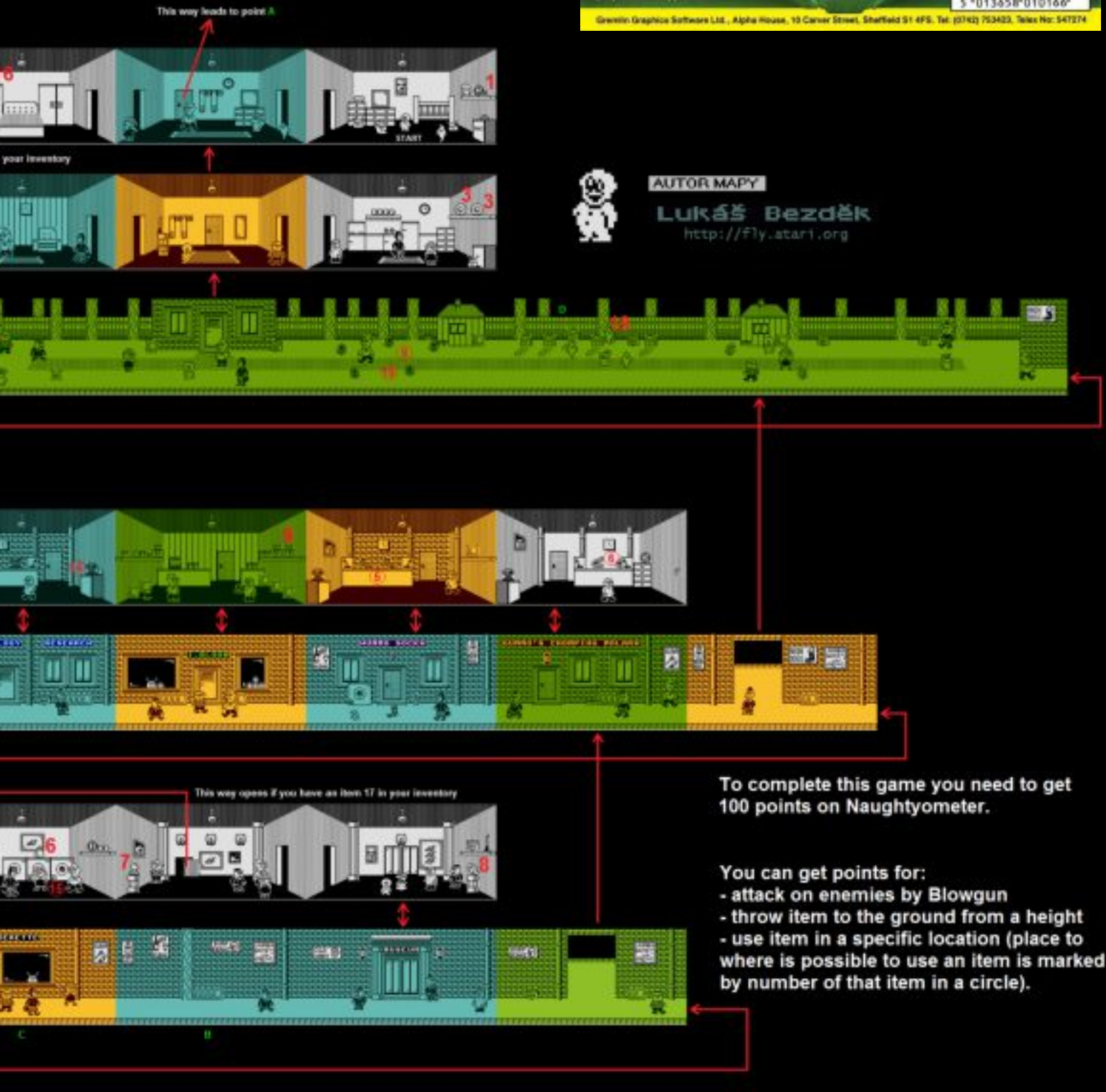
1 - Blowgun
2 - Teddy bear
3 - Plate
4 - Battery
5 - Weight
6 - Glue
7 - Bust
8 - Dummy
9 - Poison
10 - Bomb
11 - Horn
12 - Monster
13 - Potty
14 - Diskette
15 - Washing powder
16 - Credit card
17 - Key
18 - Fertilizer



E NIPPER

E MAP

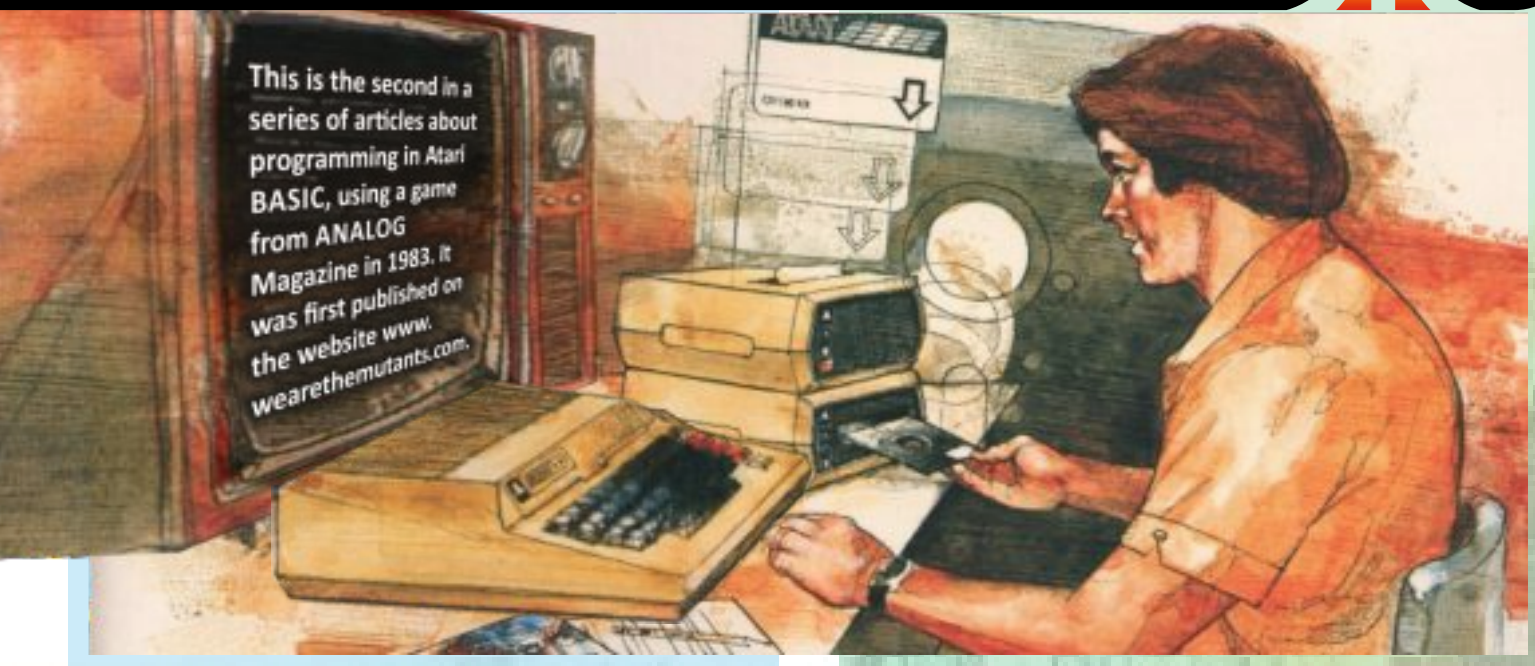




To complete this game you need to get 100 points on Naughtyometer.

You can get points for:
- attack on enemies by Blowgun
- throw item to the ground from a height
- use item in a specific location (place to where is possible to use an item is marked by number of that item in a circle).

Adventures in Atari Programming by Michael Walters



80 PRINT "Draw, Erase, Repeat"

Last time we finished drawing Space Assault's playfield, complete with our seven Vegetation Processing Plants that we must protect with the Fission Gun Tower. The planet Kala is strangely quiet, which can only mean the Clovis aliens are ready to attack - as soon as we learn how to get them on the screen!

By now you should have a good understanding of how to use PLOT and DRAWTO to create just about any image you like. But how do we make an image appear to move? If you wanted to animate an enemy ship moving horizontally (left to right) using PLOT and DRAWTO, you would need to draw the ship in colour, erase it by drawing it again with the playfield background colour, draw it again a few pixels to the right, erase it again, and so on, over and over again. This simple animation principle certainly works, but has the disadvantage of being somewhat slow. And if our ship travels over something we've drawn previously, that too will be erased, along with the ship. Of course, there's a better way to animate our ships using one of the best features of Atari home computers.

90 PRINT "Revolutionary Graphics"

If you take a quick look at popular Atari computer books and magazines of the 1980s, you'll soon discover article after article on Player/Missile Graphics, which was Atari's name for what are commonly called sprites today. The Atari 400/800 was one of the first home computers to bring this powerful capability into the hands of amateur game programmers. As an Atari aficionado, the Player/Missile Graphics feature was something you could brag about to your TRS-80 and Apple II friends. Because it was so advanced (and unfortunately somewhat arcane), entire books were devoted to the topic. What makes Player/Missile Graphics so revolutionary? First, sprites such as enemy ships, tanks, or robots are implemented as an overlay on top of the playfield. Much like cartoon animation uses transparent cels to overlay a character onto a painted background, Atari's "players" basically float over the playfield we worked so hard to draw. This means that when they move across the screen, the playfield is totally unaffected. Next, sprites are "memory-mapped" to the screen, which works so much faster than using PLOT and DRAWTO. Essentially, after a sprite image is created in memory, it can be instantly displayed on screen. We'll learn more about this in a moment.

The list of Player/Missile Graphics benefits goes on and on. We can work with four player sprites (plus smaller "missile" sprites) simultaneously, each with its own color. We can access automatic "collision detection" to immediately know when a player has touched another player or something on the playfield. We can even specify "priority" so a player can appear to move over or behind something drawn on the playfield, providing a feeling of depth. Amateur programmers who read about these powerful features in the 1980s couldn't wait to learn how to use them in their own projects.

Player/Missile Graphics started conceptually with the venerable Atari 2600 or VCS, whose hardware was designed to support popular arcade games of the era. The 2600 included two player sprites and missiles (since there were two joystick ports), plus a unique "ball" sprite for Pong-style games. When the Atari home computer hardware was designed, these graphics capabilities were improved and divided between two custom chips. Playfield graphics like the ones we've been drawing are handled by the ANTIC chip, and the GTIA chip overlays sprites on the playfield.

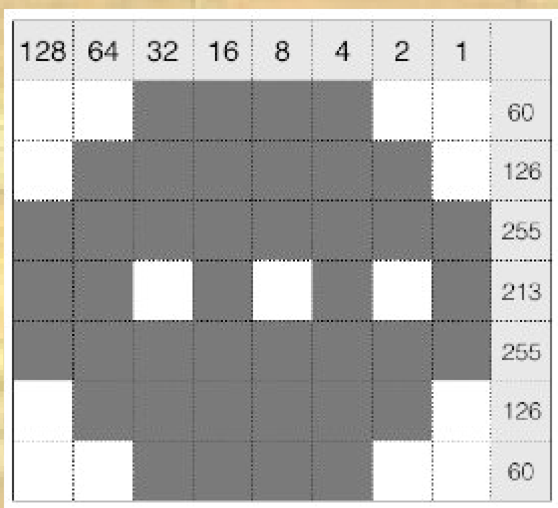
Unfortunately, Atari BASIC includes no built-in statements to use Player/Missile Graphics. This means that although the computer hardware is capable of flying our enemy ships across the screen, we don't have an easy way to tell the computer to do it. Instead, we have to directly manipulate specific parts of computer memory, and that's why Player/Missile Graphics was a somewhat mysterious topic and a bit daunting to novice programmers. I'm going to explain the

Player/Missile Graphics methods used in the Space Assault source code as simply as possible, without getting too overwhelmed in details. In fact, we won't even learn how to activate Player/Missile Graphics until our next lesson. For now, let's learn how to build those deadly Clovis ships.

100 PRINT "Building with Binary"

As mentioned earlier, player images are created in memory and "mapped" to the screen. Rather than drawing the player with PLOT and DRAWTO statements, we need to specify what the pixels of our player should look like as bytes in memory, starting with the binary number for each row of pixels. A byte contains 8 bits, so our player can be 8 pixels wide. To begin, it's helpful to sketch out the design on graph paper, so let's take a look at the first enemy ship, a traditional flying saucer shape.

Ea ch sq ua re of th e gra ph pa pe r re pr es en ts a bit , wh ich ca n be fill ed or e m pt y (1

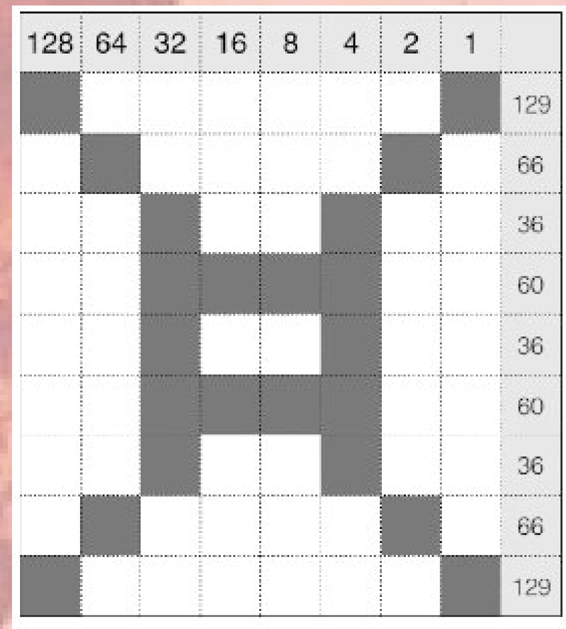


or 0). The first row of our saucer represented as a binary number is 00111100, since the first two squares are empty (0), then four squares are filled (1), followed by two more empty squares. To store these binary numbers in memory, we have to convert them to decimal numbers, which isn't so bad. Take a look at the header row with the values 128, 64, 32, 16, 8, 4, 2, and 1. By adding up these header values for the filled pixels, you're actually converting a binary (base 2) number to a decimal (base 10) number. You may notice that if all of the squares in a row are empty, the decimal number is 0, and if all of them are filled, the number is 255, which makes sense because a byte by definition may contain the values 0 through 255. Converting each row of our saucer provides the decimal numbers shown in the rightmost column above. You could also use any of several binary-decimal converter websites, but I always managed using the above method as a 16-year-old with no internet!

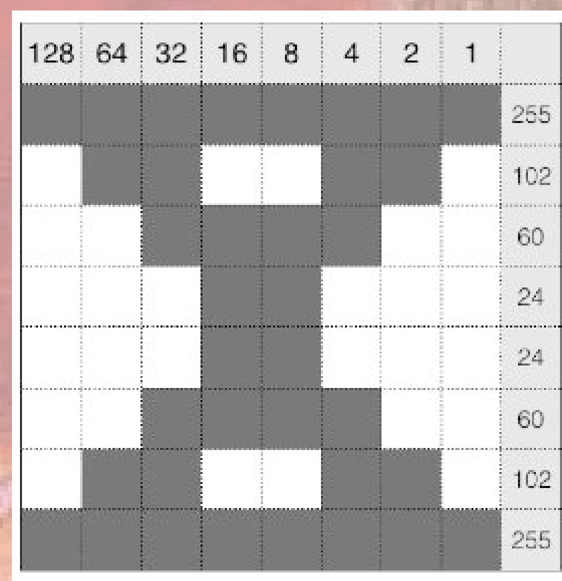
When these numbers are put into memory, the computer sees them as the binary image of our saucer and can display it instantly, without any PLOT or DRAWTO commands to draw the saucer shape or window details. Instead, the player is literally "mapped" onto the screen. This is how Player/Missile Graphics can be so much faster than the "draw, erase, repeat" method, since it's actually much less work for the computer.

Let's look at our other enemy ships. The next one is a simple satellite design.

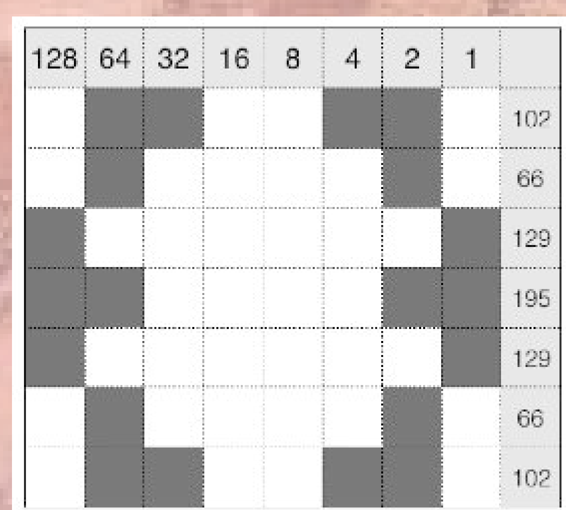
Note how this player is taller than our saucer. Although players are only one byte (8 pixels) wide, they can be



any height, including the full vertical size of the screen. Our last enemy ship was inspired by the Cylon Basestar from the original Battlestar Galactica.



The last player image is for the crosshair that determines where the Fission Gun Tower will fire.



110 PRINT "Top Secret Data"

Now that we have the numbers that define all of our player images, it's time to get them into memory. Let's finally look at some code and learn some new Atari BASIC statements.

Our new statements are READ, DATA, and RESTORE,

```
1610 RESTORE 1650:FOR N=1 TO 7:READ A:
P0$(N)=CHR$(A):NEXT N
1620 FOR N=1 TO 7:READ A:P1$(N)=CHR$(A
):NEXT N
1630 FOR N=1 TO 9:READ A:P2$(N)=CHR$(A
):NEXT N
1640 FOR N=1 TO 8:READ A:P3$(N)=CHR$(A
):NEXT N
1650 DATA 102,66,129,195,129,66,102,60
,126,255,213,255,126,60,129,66,36
1660 DATA 60,36,60,36,66,129,255,102,6
0,24,24,60,102,255
```

which are used together to work with sets of arbitrary numbers for a variety of purposes. In this case, the numbers are the decimal values we've computed that define our player images (lines 1650-1660). The DATA statement just keeps a list of numbers ready to be accessed, and the READ statement will get the next number in the list and store it in a specified variable. Since DATA statements can appear anywhere in a program, the RESTORE statement tells the program what line number to start with for the next READ statement.

Many program listings that appeared in computer magazines of this era were full of DATA statements. They were incredibly difficult and tedious to type, and it was easy to make mistakes, which often caused the program to fail. Every young programmer I knew had spent hours with a friend or parent who dictated long lists of numbers to make the process easier and less prone to error. Various magazines even invented their own verification programs, which would essentially check over your code to help you find typing errors. Thankfully, Space Assault's DATA statements aren't nearly as complex, relatively speaking.

After specifying where our numbers are with RESTORE, we use a FOR loop for each player, paying attention to the height of the player to loop through the correct number of bytes. The next number in our DATA is READ into variable A, and then stored in a separate variable for each player. These are string variables that we'll explain in the next lesson, and for now we only need to know that our crosshair image is loaded into a variable called P0$, then our saucer is loaded into P1$, our satellite into P2$, and finally our Basestar into P3$. Those variables are now in memory, so they can be "memory-mapped" to the screen once we activate Player/Missile Graphics.

You might notice that our players look a little wide. This

is because the code that we'll see in the next lesson is using another handy feature of Player/Missile Graphics that can automatically double or quadruple a player's width. For Space Assault, I used the double width option to make the aliens ships somewhat larger and easier to hit.

120 PRINT "New Invaders"

For this lesson's variations on the original code, I couldn't resist changing the alien ships into simple versions of Space Invaders shapes. I also tweaked the crosshair design.

Using the computed values from these images, along with some new colors that look better on our revised playfield, our players have quite the new look!

130 PRINT "Memory Lane"

We've learned about the Atari's revolutionary Player/Missile Graphics and spent some time designing Space Assault's players (or sprites), including enemy ships and the crosshair of the Fission Gun Tower. Now it's time to make our players move, but, if you remember, Atari BASIC includes no built-in commands to harness the power of Player/Missile Graphics. However, it does include two extremely useful tools that will get the job done: PEEK and POKE.

Quite simply, the PEEK command looks at a specific location of computer memory, and the POKE command allows us to change the contents of a location. Think of memory as simply a long series of bytes, each numbered with an address. The command PEEK 100 will return the value stored in memory address 100, and the command POKE 100, 255 would store the value 255 into that address. This sounds a bit mundane, but when you realize that everything the computer does is based on the contents of memory, you'll see that anything is possible using PEEK and POKE.

The true usefulness of manipulating memory is found in changing the values of registers. A register is no different from any other memory address, except that the Atari computer constantly monitors the values stored in registers, and changes in these values make things happen. Actually, some of the BASIC commands we have learned are really just a POKE in disguise. For example, we've seen that the SETCOLOR command specifies a color for use with PLOT and DRAWTO. The command SETCOLOR 4,0,0 sets the background color of the playfield to black (4 means background, and 0,0 are hue and luminance values). When this command is executed, what really happens is POKE 712,0. This is because memory address 712 is the "background color" register. The Atari hardware continuously looks at the value in register 712, and changes the background color of the playfield accordingly. The SETCOLOR command only exists to make it easier for BASIC programmers, so they won't have to remember that 712 is the register, or do the math to convert hue and luminance values into a single byte.

As you might imagine, there are several registers

dedicated to controlling Player/Missile Graphics, and since Atari BASIC doesn't provide any commands like SETCOLOR for these registers, we have to learn their memory addresses and POKE them ourselves. Although PEEK and POKE commands were a staple of most home computer BASIC programs in the 1980s, today's programming environments make a tremendous effort to prevent anything resembling direct memory manipulation. Since modern operating systems allow several programs or processes to operate simultaneously, it's important to prevent one program from "stepping on" another. PEEK and POKE would be a hacker's dream if protected memory management didn't exist today.

140 PRINT "String Theory"

Before we start POKEing, we need to make a short detour to discuss "string" variables. A string is simply a text variable, meaning it represents a series of characters rather than a numerical value. String variables are notated with a dollar sign, such as A$ or NAME$. In Atari BASIC, string variables have a fixed size that is specified with the DIM statement (short for "dimension"). For example, DIM A$(10) means that the string variable A$ can contain 10 characters. Each character of a string can be referenced by using an index, so A$(3) refers to the third character of A$.

Looking back at Lesson Three, after figuring out the byte values for each row of our player images, we used a FOR loop to READ each byte from our DATA statements, and then used a function called CHR$ to convert each of our bytes into a character, which was assigned to an index of a particular string.

```
1620 FOR N=1 TO 7:READ A:P1$(N)=CHR$(A
):NEXT N
```

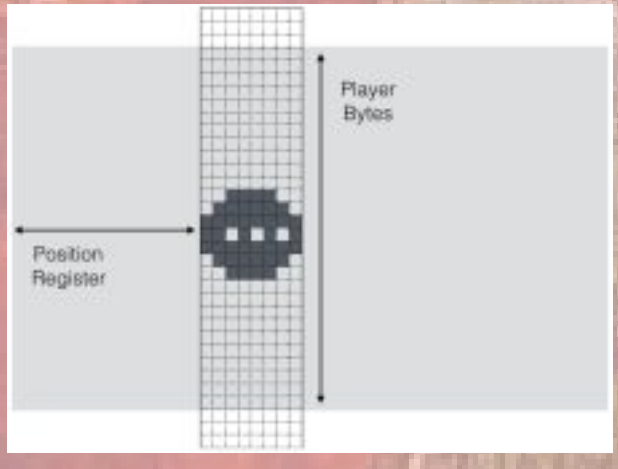
The CHR$ function returns the character represented by a specific number in the ATASCII character set (Atari's extended version of standard ASCII). We don't really care what these characters are, but we need to convert our bytes into characters since string variables only contain text. Why in the world have we done this? Again, it's all about memory. When programming Player/Missile Graphics, the memory location of each player's image data must be known. When we DIM a string variable, the Atari sets aside a location in free memory that's just the size we need, and we can now obtain that location by using the ADR statement. For example, ADR(A$) will tell us the address of the memory containing the bytes of A$. So, if our player image is stored in a string variable called P1$ (as in line 1620 above), then we can tell the computer that our player image is located in memory at ADR(P1$). Let's take a look at the first few lines of our Player/Missile Graphics initialization.

```
1490 REM * Initialize & P/M Setup
1500 TRAP 1510:DIM BUILDING(7),XPOS(15
),YPOS(15),PMMOV$(100),P0$(7),P1$(7),P
2$(7),P3$(7)
1510 X=125:Y=70:SC=0:HIT$=0:MOVE=ADR(P
MMOV$):PM0=ADR(P0$):PM1=ADR(P1$):PM2=A
DR(P2$):PM3=ADR(P3$)
```

You'll notice that we can include multiple variables in one DIM statement, so we first specify several string variables (line 1500). Note that we have four player strings: P0$, P1$, P2$ and P3$. Each of these are sized with the height of the player images we designed in Lesson Three. Next, we store the memory address of each of these strings into variables called PM0, PM1, PM2 and PM3 (line 1510). Now we can easily tell the Atari where the image bytes of each of our four players (three alien ships and our crosshair) are in memory.

150 PRINT "Moving Through Memory"

Before continuing, we need to discuss how the Atari actually determines where a player is displayed on the screen. Another topic we mentioned in Lesson Three was "memory mapping," meaning that players are directly mapped onto the screen from their binary representation. The location of a player is controlled by two factors: the value of a horizontal position register, and the spot where the player's bytes exist in a consecutive area of memory.



For each player, a series of consecutive memory addresses is reserved that construct a kind of "strip" from the top to the bottom of the screen. If the player bytes are located at the beginning of the strip, it will be mapped to the top of the screen, and if the player bytes are located at the end of the strip, it will be mapped to the bottom of the screen. Actually, this strip extends past the visible screen area, so players can move off screen where they are no longer visible.

The player's horizontal position is stored in a dedicated register for each player. Simply POKEing a value into this register will instantly pop the player to a new position, making horizontal movement extremely easy. Vertical movement is a bit more difficult, since we need to copy our player bytes into a different location along the memory strip, which we'll discuss in a moment.

160 PRINT "Initializing Player/Missile Graphics"

It's finally time to fire up Player/Missile Graphics by POKEing all of the necessary registers. Our first task is to tell the Atari what part of memory to use for the player memory strips we just described. To do this, we need to find out where some free memory exists, which we can do by PEEKing into register 145, where the Atari stores the beginning address of free memory.

```
1670 PMBASE=INT((PEEK(145)+3)/4)*4:POK
E 54279,PMBASE:PMB=PMBASE*256
1680 POKE 704,10:POKE 705,216:POKE 706
,104:POKE 707,56
1690 POKE 623,1:POKE 53257,1:POKE 5325
8,1:POKE 53259,1:POKE 559,46:POKE 5327
7,3:GOSUB 1280
```

After performing some unusual math on the value of PEEK(145), we store this location in a variable called PMBASE (line 1670), the base address of our player memory strips. Truthfully, I never fully understood the formula being used here, since I copied it from other articles I read when I was learning Atari BASIC as a teenager. As I mentioned in Lesson Three, my goal is to explain the concepts of Player/Missile Graphics without getting stuck on elements that I'm sure scared away other young programmers at the time. The important thing is understanding why we need this memory address for the player memory strips.

The next step is to assign a color to each player, which is POKEd into registers 704 through 707 (line 1680). These values are computed by using a simple formula, multiplying a hue value by 16 and adding a luminance value.

Next, we specify our player "priority" so that our players will be drawn over the playfield (register 623). Other values for this register could allow players to move underneath the playfield, which would look funny against our starry sky, but would look great if we wanted our ships to move behind clouds, for example. Then we specify that each of our players should be double-width (registers 53257-53259), which makes them a little easier to hit with our Fission Gun, as well as double height (register 559). Finally, Player/Missile Graphics are officially activated by POKEing register 53277.

At this point you may wonder how any amateur programmer ever learned to work with Player/Missile Graphics at all. How could anyone know all of these register addresses? For myself, the answer is that I read every scrap of documentation I could find in computer magazines and books, and then tried things out even if I didn't fully understand what was going on. Even without the internet, there was a huge amount of sample code available to study, and the more I experimented, the more I settled into my own boilerplate way of initializing and working with Player/Missile Graphics. Today, nearly all of this knowledge is just a Google away, including Atari's own official documentation, De Re Atari. This book was practically a magical tome to amateur Atari programmers in the 1980s, and it's still fascinating to peruse today.

170 PRINT "Beyond BASIC"

We've just learned that moving a player horizontally is easy, thanks to the horizontal position register. You can imagine a simple loop that POKEs the loop variable into the horizontal position register, gliding our player smoothly across the screen at the same "altitude," just like the bonus saucer in Space Invaders.

```
8000 FOR X=48 TO 198:POKE 53249,X:NEXT
 X
```

This sample code (not part of the Space Assault source) will move our saucer (Player 1) from left to right across the screen, assuming our player bytes are positioned in a visible portion of our player memory strip. Player 1's horizontal position is stored in register 53249, and the player is instantly moved when the value of that register changes. By the way, horizontal values below 48 and more than 198 are "off screen," so moving the player beyond these limits would make it invisible.

But moving a player vertically requires a bit more work, since the bytes that make up the player image must be copied into new addresses along our player memory strip. If you think about it, this is similar to the "draw, erase, repeat" method we discussed in Lesson Three, but much less time-consuming, since we don't have to worry about PLOT and DRAWTO commands, thanks to Player/Missile Graphics memory mapping.

Unfortunately, even though this process is relatively quick, Atari BASIC can't perform these operations quickly enough for a video game. Remember that Atari BASIC is an interpreted language, so our commands must be deciphered "on the fly" before the actual task is executed. The only solution to this problem is to bypass the BASIC interpreter, using the actual machine language that the 6502 processor in our Atari understands. Fortunately, when I was programming Space Assault as a teenager, I discovered an article in ANALOG Computing Issue 10 by Tom Hudson that provided exactly the code I needed to make my game faster. This code bundled up everything necessary to position any player both horizontally and vertically at the same time, and essentially extended Atari BASIC with a new, extremely useful function.

Atari BASIC was designed with a "hook" that makes this extension possible: the USR statement (short for "user routine"). After loading machine language code into a particular location in memory, Atari BASIC can use the USR statement to tell the Atari to stop interpreting BASIC, execute the code at the specified location, and then return to BASIC and pick up where we left off. This forward thinking feature allowed programmers to perform powerful tasks without making the full jump into the somewhat difficult world of machine language.

The player movement machine language routine, provided as a series of bytes in a DATA statement in the ANALOG article, is loaded into a string variable, exactly like we did with our player images.

```
1560 RESTORE 1570:FOR N=1 TO 100:READ
A:PMMOV$(N)=CHR$(A):NEXT N
```

The address of this string, which is ADR(PMMOV$), is then stored in a variable called MOVE (line 1510 above), which we will use to move our players in a

single line of code. Let's modify our sample code from above to add some vertical movement.

```
9000 Y=12:FOR X=48 TO 198
9010 IF X/2=INT(X/2) THEN Y=Y+1
9020 A=USR(MOVE,1,PMB,PM1,X,Y,7)
9030 NEXT X
```

We've added another variable for our vertical position, which is increased as the loop executes, moving our player from the top left to the bottom right of the screen. Our vertical variable is only increased every other time through the loop (using the method described in Lesson Two to draw our checkerboard pattern), because the vertical size of the screen is shorter than the horizontal length.

All of the work is done in the USR statement, which passes a series of variables to our machine language routine (line 9020). We include the address of the machine language routine itself (MOVE), the number of the player to move (1), the address of our player memory strips (PMB), the address of our player image bytes (PM1), our horizontal and vertical positions (X, Y), and finally the height of our player (7). Remember that the machine language routine also takes care of the horizontal position register, so we no longer have to POKE that ourselves. The variable A that is returned by the USR statement is ignored in this case, but other kinds of routines (such as advanced math functions) could return a result this way.

It's interesting to note that rival BASICs available for the Atari later included built-in commands to perform memory movement quite similar to this routine. Atari Microsoft BASIC II included a MOVE statement that worked in much the same way and could have been used for speedy player movement.

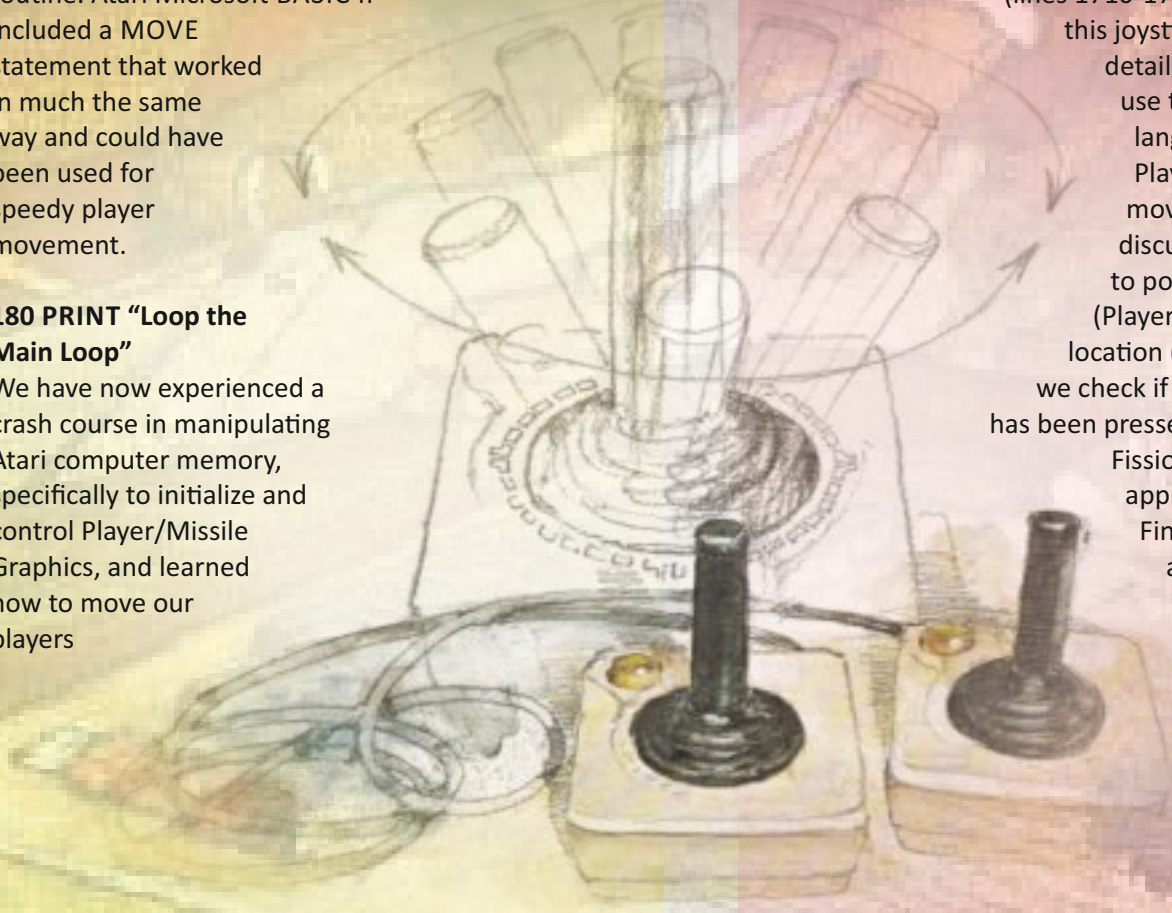
180 PRINT "Loop the Main Loop"

We have now experienced a crash course in manipulating Atari computer memory, specifically to initialize and control Player/Missile Graphics, and learned how to move our players (sprites of Space Assault's enemy ships and crosshair) across the playfield. With those powerful techniques under our belt, this lesson is action-packed! Not only will we cover joystick control, but also how to fire the mighty Fission Gun Tower. To get started, let's look at the heart of the game, often called the "main loop."

In nearly all game programs, there's a series of actions that must occur seemingly simultaneously. In Space Assault, the crosshair must be moved based on joystick movement, enemy ships must travel towards their intended target, explosions and sounds should occur accordingly, and so on. However, in the 8-bit home computer era, processors like the 6502 in Atari home computers were single-threaded, meaning they performed a single task, then another task, and so on. But if these game actions simply "take turns" very quickly inside a loop, they appear to happen at once. Let's examine the structure of Space Assault's main loop, keeping in mind that some concepts and statements will be explained later in this lesson.

```
1700 REM * Main Loop
1710 ST=STICK(0)
1720 X=X+XPOS(ST):Y=Y+YPOS(ST)
1730 IF X<48 THEN X=48
1740 IF X>198 THEN X=198
1750 IF Y<12 THEN Y=12
1760 IF Y>50 THEN Y=50
1770 A=USR(MOVE,0,PMB,PM0,X,Y,7)
1780 IF STRIG(0)=0 THEN GOSUB 1380
1790 GOSUB 1100:IF EXP=1 THEN GOSUB 14
60
1800 GOTO 1710
```

First, we handle the "human" turn by reading the joystick and determining the next horizontal and vertical coordinates for the crosshair (lines 1710-1760). We'll look at this joystick code in more detail shortly. Next, we use the machine language Player/Missile Graphics movement routine discussed in Lesson Four to position the crosshair (Player 0) in the correct location (line 1770). Then we check if the joystick trigger has been pressed and fire the Fission Gun Tower if appropriate (line 1780). Finally, the enemy gets a turn, along with a check to see if an enemy explosion is in



progress (line 1790). Then it all starts over again by jumping back to the start of the main loop (line 1800). This sequence of events will occur over and over again until the game is over, which means all of our bases have been destroyed.

In most modern programming, the concept of a "main loop" has been lost, since so many tasks are automatically handled by sophisticated operating systems and programming frameworks. Even early programmers of the Apple Macintosh in 1984 didn't have to write code to move the mouse pointer in their applications, since the operating system was in charge of this task. Most programming today is event-driven, meaning the programmer only writes code to handle events (such as the user clicking a button, or a web service sending a message), while the foundation that generates these events and weaves them together is out of the programmer's domain. In addition, today's hardware with multiple processors truly can perform tasks simultaneously, so a programmer could write a thread to handle crosshair movement that operated independently of a thread handling enemy ship movement, for example.

190 PRINT "Taking Direction"

Since the Atari joystick was so well known due to the success of the Atari 2600 (VCS), the designers of Atari BASIC made certain it was easy for programmers to use on Atari home computers. The STICK statement allows us to read which direction the joystick is pressed, and the STRIG statement tells us if the joystick trigger is pressed. Both of these statements are deciphered using a single number from 0 to 3 corresponding to the four joystick ports, so STICK(0) will return the direction of the joystick in the first port.

The values returned by STICK are somewhat unusual, but can be easily deciphered using the diagram below.

For example, if STICK(0) returns 7, we know the joystick has been pressed right. In Space Assault, we keep track of the position of the crosshair using two variables: X for the horizontal position and Y for the vertical position. If the joystick is pressed right or left, we need to make X larger or smaller, and the same is true for up and down and Y. For diagonal movement, we increment or decrement both X and Y accordingly. This could be done easily with eight distinct IF-THEN statements, such as IF STICK(0)=7 THEN X=X+1. But remember, these eight statements will be executed each time through the main loop, potentially making the game slower (once again, thanks to Atari BASIC being an interpreted language). Let's look at a technique using arrays to make our joystick code smaller and more efficient.

```
1540 RESTORE 1550:FOR N=1 TO 15:READ A
:XPOS(N)=A:READ A:YPOS(N)=A:NEXT N
1550 DATA 0,0,0,0,0,0,0,0,4,4,4,-4,4,0
,0,0,-4,4,-4,-4,-4,0,0,0,0,4,0,-4,0,0
```

During Space Assault's initialization, two arrays called XPOS and YPOS are filled with 15 numbers each. If you count the position of the non-zero numbers in the array, you'll see that XPOS(7) is 4 and XPOS(11) is -4, for example. With these two arrays, our main loop joystick code can be reduced to simply adding the array value at the index corresponding to STICK for both X and Y. Remember that adding a negative number is the same as subtraction.

```
1710 ST=STICK(0)
1720 X=X+XPOS(ST):Y=Y+YPOS(ST)
```

You might be wondering why X and Y are changed by 4 rather than 1, and the reason is that it makes the game appear faster. Moving the crosshair by one pixel at a time seemed to be slow, and moving by four pixels seemed to provide a good balance between speed and the "jumpiness" that would occur if the increment was too large. This is another good example of a programming compromise, making Space Assault performance seem better while sacrificing some animation smoothness.

Unfortunately, after cleverly avoiding eight IF-THEN statements with the joystick array technique, we next use four IF-THEN statements to keep our crosshair bounded to the playfield so it can't be moved offscreen (lines 1730-1760 above). I mentioned in Lesson Four that a player's onscreen horizontal position was roughly from 48 through 198, so X cannot be larger or less than these values, meaning the crosshair is constrained to the same horizontal position. The same goes for Y with a minimum vertical position of 12, although we keep the maximum position at 50, which forces the crosshair to remain in the upper half of the playfield where the enemy ships move. I'm certain these coordinate bounds could have been handled more efficiently, but the existing code was good enough for me as a teenager.

Interestingly, a reader letter was included in ANALOG Computing Issue 15 (two issues after Space Assault

was published) describing how the reader made the game seem faster by increasing the speed of movement even more than I did. He did this by doubling the values of the XPOS as YPOS arrays, so X and Y are changed by 8 pixels with each movement. This is way too jumpy for my taste, but I'm glad the reader was able to customize the game to his liking. He also cleverly made the alien ships move faster as the game progresses, essentially adding difficulty levels, something I didn't bother to do at the time. This kind of user modification was a truly wonderful aspect of programming in the early 1980s, because everyone was learning together via resources like source code listings in magazines.

200 PRINT "Drawing the Blast"

As we saw above, every time our main loop is executed, we use the STRIG command to check the joystick trigger. We need to execute code to fire the Fission Gun Tower if the trigger has been pressed, but it's best to keep the lines of code associated with this out of the main loop so the loop will run faster. Since firing the gun is something that will be done many times during game play, this is a good time to use a subroutine. Atari BASIC supports using subroutines with two statements. The GOSUB statement will immediately jump to a specified line of code, while remembering where the program was executing before the jump. This is similar to the GOTO statement, except GOTO doesn't keep track before jumping. The RETURN statement will conveniently jump back to where the last GOSUB was executed, making it easy to call a section of code and pick up where we left off when our subroutine is finished.

When the joystick trigger is pressed, our main loop jumps to the "Shoot Tower Gun" subroutine (line 1780 above).

```
1370 REM * Shoot Tower Gun
1380 POKE 53278,0
1390 COLOR 2:PLOT 150,63:DRAWTO X-44,Y
-12:FOR N=1 TO 10:SOUND 0,N,12,8:NEXT
N:SOUND 0,0,0,0
1400 COLOR 0:PLOT 150,63:DRAWTO X-44,Y
-12
```

Our main task in firing the gun is drawing the beam from the tip of the tower to the current position of the crosshair controlled by the joystick. First we select our purple color and PLOT the tip of the gun, and then DRAWTO the crosshair using the X and Y coordinate variables that represent the position of the crosshair (line 1390). We have to do a little conversion between Player/Missile Graphics coordinates and playfield coordinates, since the number of pixels is slightly different between the two. This conversion also takes into account the shape of the crosshair player, so the line we draw will end roughly in the center of the crosshair. After drawing the beam, we create a short sound effect using a FOR loop and the SOUND statement. We'll cover the details of Atari BASIC sound in a future lesson.

The reason we can easily use our X and Y variables in this subroutine is because all variables in Atari BASIC are global, meaning they can be referenced anywhere in our program. Since the creation of BASIC, programming languages have increasingly emphasized the concept of variable scope, meaning a block of code should only reference variables declared in the same block or variables that have been passed to the block from the calling code. BASIC's global variables are an easy concept to learn for a beginning programmer, but would almost never be used in modern programming languages today.

Since we now have a purple line drawn across the playfield, the next step is to erase it so our sky won't be permanently filled with beams. All we need to do is switch to the background color and PLOT and DRAWTO the identical line, effectively erasing the beam (line 1400). At this point we've blasted our Fission Gun Tower, but how do we know if we've hit an enemy ship? We can use another powerful feature of Player/Missile Graphics to find out.

210 PRINT "Detecting Collisions"

In almost any computer game, it's important to know when objects on screen are "touching." In Pong, for example, we need to know when the ball has touched a paddle. For Space Assault, we need to know that the beam we've just drawn has touched, or collided, with an enemy ship. This is called collision detection, and Player/Missile Graphics handles this automatically via several registers waiting for us to PEEK. Each player has dedicated registers that indicate if the player has touched another player, or if the player has touched something drawn on the playfield. The convenience to the programmer of automatic collision detection cannot be overstated, and it's another huge advantage of using sprites.

Notice that before we drew our beam, we POKED 0 into address 53278 (line 1380 above). This clears all collision registers so they are reset and ready to be checked at the appropriate time. After drawing the

beam, we can PEEK into the appropriate registers depending on which kind of enemy ship is traveling across the screen, indicated by the variable SHIP.

```
1410 IF SHIP=1 THEN IF PEEK(53253)<>0
AND PEEK(53261)<>0 THEN EXP=1:SD=40:SC
=SC+500:POKE 53249,0:GOSUB 1280
1420 IF SHIP=2 THEN IF PEEK(53254)<>0
AND PEEK(53262)<>0 THEN EXP=1:SD=40:SC
=SC+100:POKE 53250,0:GOSUB 1280
1430 IF SHIP=3 THEN IF PEEK(53255)<>0
AND PEEK(53263)<>0 THEN EXP=1:SD=40:SC
=SC+300:POKE 53251,0:GOSUB 1280
1440 RETURN
```

If the Combat Saucer (SHIP=1) is attacking, we need to look at the collision registers for Player 1 (line 1410). PEEKing address 53253 tells us if Player 1 has touched something drawn on the playfield, which would be the line we drew to represent the beam. PEEKing address 53261 tells us if Player 1 has touched another player, which would be our crosshair. The reason we check both of these registers is to verify a "direct hit," meaning the crosshair was positioned on the enemy ship when the gun was fired. If we didn't check the player-to-player collision register, then hits could be scored by firing "past" the enemy ship, since the beam would be drawn through the player.

If both of these registers have a non-zero value, the enemy ship is hit! After setting some variables that we'll discuss in a moment and increasing the score by the value appropriate to the type of destroyed ship, we POKE the appropriate horizontal position register (53249 for Player 1) with 0 to instantly move the enemy ship offscreen, causing it to disappear. And finally, we use GOSUB to jump to a subroutine that selects a new enemy ship to terrorize the planet. All of these steps are duplicated for the three kinds of enemy ships, and the final RETURN statement jumps back into our main loop.

220 PRINT "Explosion Multitasking"

It may be obvious by examining the linear nature of Space Assault's code, but it's important to realize that Atari BASIC executes synchronously. As we discussed earlier, only one task happens at a time, and the concept of asynchronous processing that's so common today was simply not available to amateur programmers in the early '80s. However, the appearance of more than one task happening at once was possible with clever programming techniques. Space Assault attempts this in a somewhat primitive way so the main loop can continue to cycle during an explosion effect after an enemy ship is hit. This allows the game player to keep moving the crosshair with the joystick while the explosion is occurring.

In our "Shoot Tower Gun" subroutine, after a successful hit we set EXP to 1 to indicate an explosion is in progress, and set SD (short for "sound") to 40 (line 1410 above). Every time we proceed through our main loop, we check EXP to see if we should call the "Explosion Routine" (line 1790 above).

```
1450 REM * Explosion Routine
1460 SD=SD+5:IF SD>100 THEN SOUND 1,0,
0,0:EXP=0:RETURN
1470 SETCOLOR 4,0,14:SETCOLOR 4,0,0
1480 SOUND 1,SD,8,8:RETURN
```

This short and simple subroutine increases SD for use in a SOUND statement. If SD is greater than 100, the explosion is "over," so we stop the explosion sound and set EXP to 0 (line 1460), which will cause our main loop to quit calling the explosion subroutine. In other words, EXP is a flag variable with two states (explosion in progress or explosion not in progress). Using flag variables to control the flow of execution is a quite common programming technique.

Next we use SETCOLOR to change the playfield background color to white and quickly back to black, causing a strobe effect (line 1470). Finally, we use SD as a parameter of the SOUND statement, which makes the explosion noise progressively lower in tone as the subroutine is continually called (line 1480). And of course, the RETURN statement jumps back to our main loop where we continue reading the joystick, moving the crosshair, and keeping the game play flowing.

230 PRINT "Wrapping the Crosshair"

For this lesson's variations on the original code, I decided to add horizontal wrapping when the crosshair is moved to the extreme left or right of the screen. By changing two lines of our maximum bounds check code in the main loop, it's quite simple to swap the X coordinates to the opposite extreme when the limit is reached, instantly popping the crosshair to the other side of the playfield. I've also added a small sound effect when this happens, which is a fast rising tone ("whoop") when wrapping to the right side, or a fast falling tone when wrapping to the left.

```
1730 IF X<48 THEN X=198:FOR N=1 TO 20
STEP 2:SOUND 0,N,10,8:NEXT N:SOUND 0,0
,0,0
1740 IF X>198 THEN X=48:FOR N=20 TO 1
STEP -2:SOUND 0,N,10,8:NEXT N:SOUND 0,
0,0,0
```

Vertical wrapping is also possible, but doesn't really make sense for Space Assault, since the enemy ships always stay in a somewhat narrow band at the top of the screen. With the addition of crosshair wrapping, another defensive technique is available to protect the planet from those evil Clovis Aliens.

Next time, the aliens get a chance to strike. We'll learn the secrets behind their movement, and, unfortunately for the planet Kala, how to destroy a Vegetation Processing Plant!


COMMANDO


II
PAL


PINK PANTHER


STAR WARS


WARHAWK


NEMESIS

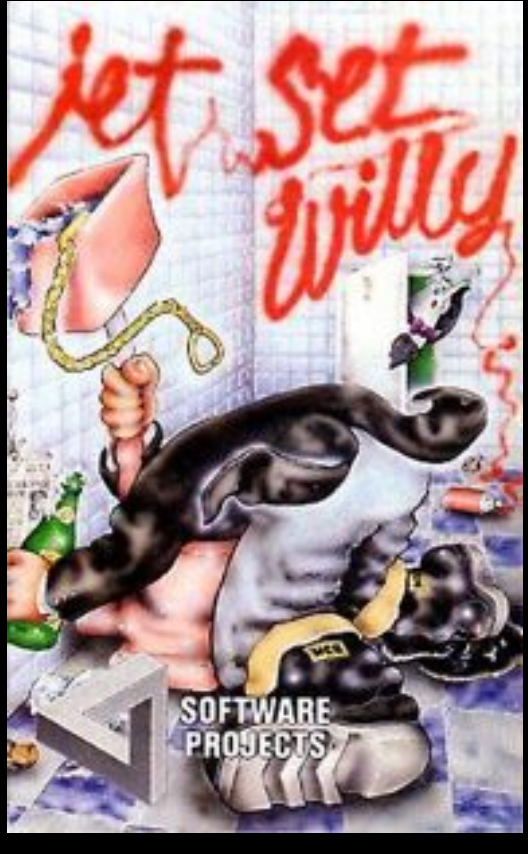

ocean

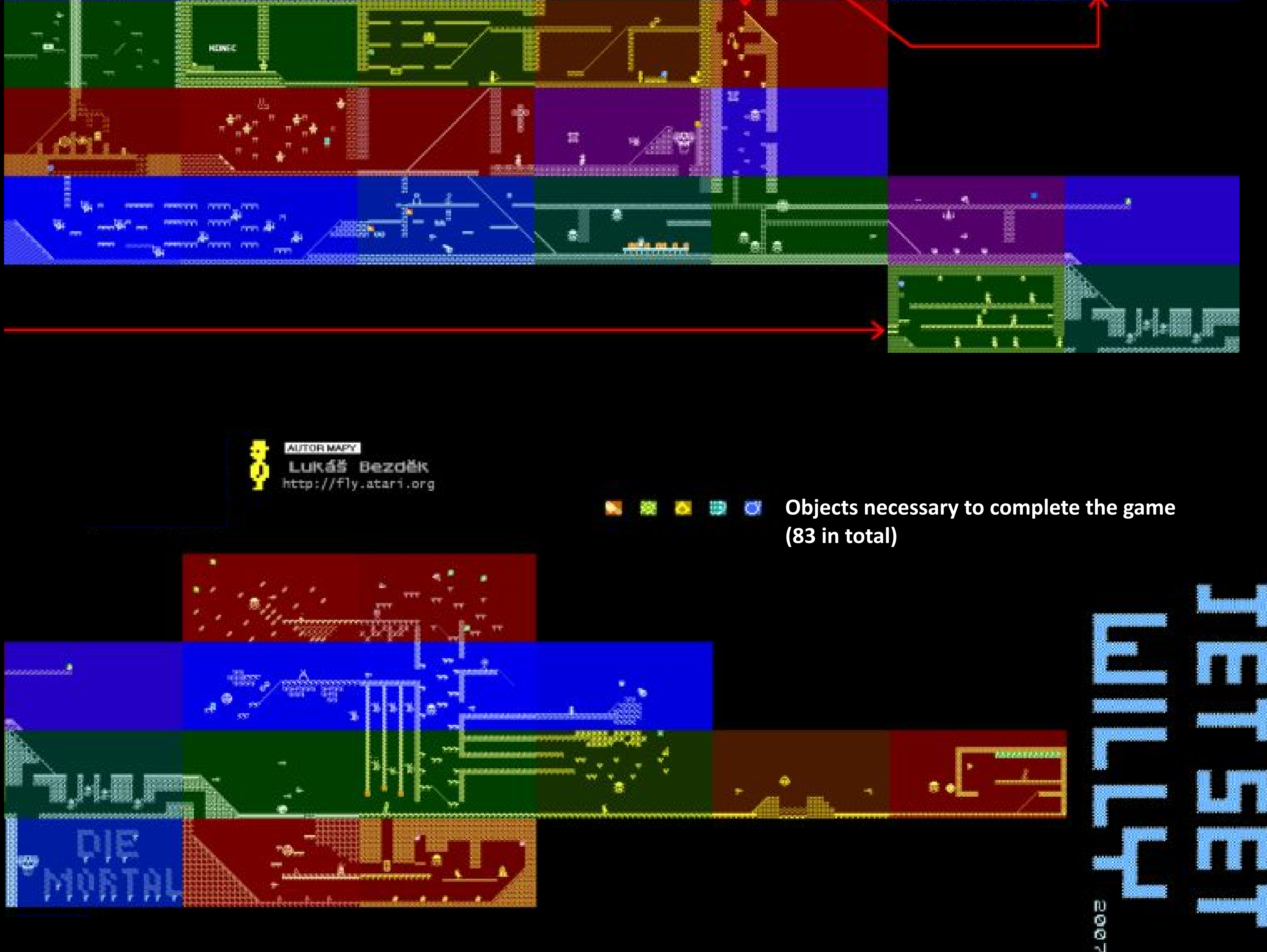

SCARY SCHOOL


RASTAN


SPELLBOUND

GAME MAP





Objects necessary to complete the game (83 in total)

CELEBRITY CORNER

Paul Daniels

By Mike Roberts

You'll like this interview, not a lot, but you'll like it!

Paul Daniels is a household name thanks to his highly entertaining magic shows and hosting the quiz game 'Odd One Out'. What isn't so well known is his long running obsession with computers.

His equipment list is amazing — enough to bring pangs of envy into any true computer freak. The main computer is an Atari 800, supported in turn by two disk drives, two printers (an Atari letter quality and an Epson FX80), a TV with a direct input monitor connection, two modems, a tape deck, and a pair of Quickshot II joysticks.

"There's also some other odds and ends lying around, touch tablets and the like" admits Paul. It's easy to see what he meant by 'lying around' for his computer room is packed with equipment and software.

In one corner is a large brown box. "That's a spare Atari 800" he says "in case this one ever breaks down. I got it for £40, which must make the 800 one of the best value computers on offer at the moment".

Thinking about it breaking down, was the Atari reliable and did he have trouble with it? "No, I've had no trouble with it at all, it's been left on for over four years you know. They don't mind being left on but dislike being turned on and off all the time".

Scattered around the room are various games piled up. What are his favourites and does he keep his high scores? "I used to keep my high scores, but I've grown up now" he says "I don't really have any favourites, I can pick up a game that I haven't played for months, even years sometimes. There is one game I have problems with and that's 'Jumbo Jet Pilot' by Thorn EMI (nb. now renamed 'Creative Sparks'), I can't even land it. I even got a real 747 pilot to try and land it and he couldn't!". If any of our readers do know how to land it could they drop us a line, and maybe we can help Paul!

Lying in the middle of one room is a BBC micro and a copy of Elite by Acornsoft: is he changing allegiance from Atari to Acorn?: "No, I got that when I was working with Acornsoft on 'The Paul Daniels Magic Box'. I didn't do any actual programming on that but I was at the initial design stage, it's very good. Detective is the best trick. I'm amazed by it and I know how it's done" (like all good magicians he's not telling anybody that!) "It's a very good program, the magician can use the computer as a prop, it doesn't do any of the actual magic".

What about 'Elite'? "I picked that up when I was wandering around in Leeds, I haven't played it yet, but I'm told that it's quite good".

"I will really be using the BBC as a Prestel terminal, but I'm having some problems" he says. "With the Atari I can just dial up any computer in the world, but when I joined Prestel I got no detail on how to connect or anything, I found out that I needed special communications software on ROM, now I find out that I need to open up the computer to install it".

Some time ago an adventure game was written by him called 'The Paul Daniels Magic Adventure', it can still be obtained from Amazon Systems Software at:

Merlewood
Lodge Hill Farm
Farnham

"It's basically a straight adventure with a lot of graphics and sound surprises".

One thing that struck me was the thing that he uses to anchor joysticks down on. "It's called a 'knee desk', I picked this up in America, but anybody can make one. It consists of a small bean bag attached to a flat board. The bean bag rest solidly on your lap when you are sitting down" this gives an excellent platform to stick down 'suction cup' joysticks such as the Quickshot II.

Next month we bring you another celebrity who is getting into the world of computers and games: Peter Lorimar, of Leeds Football Club.

STAR TURN

Paul Daniels

We all know that computers can perform magic — but what happens when a real magician gets his hands on one? Mike Gerrard went to see ace magic man and TV star, Paul Daniels, who not only enjoys playing computer games but also had a hand in writing a computer Adventure.

Magician Paul Daniels isn't a new recruit to the computer craze, as he's owned his Atari 800 for four years now and first got interested in micros a year before that.

"It seemed that every other person I got up on stage was a computer programmer and I'd no idea what that was at all. I think it's part of my job to be able to talk to as many of the audience as possible about their jobs and also I wanted to find out about computers out of interest.

"For a year I bought every magazine on the market, read everything in them and at the end I was none the wiser. I don't know about now, but then they were written by people who knew something for other people who knew something, not for the likes of me who knew nothing. You don't try to teach people French by showing them a book written entirely in French, do you? But that's what they seemed to be doing.

"Anyway, at the end of that year I simply walked into a shop and said, "Look, there's £1,000, I want a computer and I know nothing about them." The assistant said, "Have a ZX-80," and I said, 'No thanks, I know that much. I do want one with a keyboard."

"I really believe I got very lucky because that guy sold me an Atari 800 and that machine's been switched on virtually non-stop for about four years and it's fine. And I mean non-stop — I literally leave it on the whole time."

For his £1,000 Paul also acquired a disc drive, a cassette player and a few programs. In the meantime he has added an Epson MX-80 printer, a buffer that allows him to use the Atari and the printer simultaneously and a couple of modems.

"I love those . . . when you start talking to other people with micros, that's when the fun starts. One of the modems restricts me more or less to England, but with the other I've been plugged into the States and everywhere. I also do a lot of letter-writing on the Atari."

"I got into programming by typing in listings from magazines and I learned more from doing that a line at a time and discovering what each one did than from any book . . . certainly not the manual which I couldn't understand. I'm delighted with the Atari and I'd only get another micro now if I could run the house with it when I'm away . . . you know, switch the lights on and off and draw the curtains and things.

Paul loves games and had a hand in writing *Paul Daniels' Magic Adventure* which was published a few months ago.

"It's a traditional type Adventure but I also wanted it to have games sections and a few tricks in it. Someone else did the programming, but I gave him the story-line, the jokes and devised the games that are in it."

"At one point you go to a magic show, for which you need a real pack of cards. You cut the deck and then, after answering a few questions, the computer tells you what your card is. Very clever! I'm also involved in another piece of software that should be out soon which consists entirely of magic tricks on the computer written by someone in Cambridge who's a magician, a programmer and a mathematician."

We asked Paul about his favourite games.

"I tell you what I've been using more than anything lately and that's the *Atari Touch Tablet* and *Atari Artist*. It is absolutely brilliant and I sit here for hours just playing around with it . . . the shimmering rainbow effect you can get is beautiful."

"My all-time favourite joystick game, though, is one called *Sea Dragon* by Russ Wetmore which has been out a couple of years. You go through a series of underground passages, shooting mines and dodging things and so on. The sound and graphics are excellent and really bring the best out of the Atari. I have got through it to the end but only after hours and hours at it. I've had more fun with this than with anything. In fact, I wrenched two joysticks apart playing it which took me to designing and building my own. They're quite easy to make, really."

Paul says that his specially designed joysticks may find their way onto the market, but the details for that haven't been finalised yet. He also mentioned a software protection device that he stumbled across while designing his Adventure. "But I'm not telling you what it is!"

We should have known: magicians never reveal their secrets.

Modern Atari Peripherals by Robin Edwards

Looking for a (late) Christmas gift for your Atari computer? Here's a quick round up of some of the currently available Atari 8-bit peripherals from sellers in Europe and the USA. Not bad for a computer approaching 40 years old!

Cartridges

The!Cart v2 - A multi-cart with an amazing menu system and massive 128MB of storage. The included software allows you to put your ROMs, CARs, XEXs and ATRs into a collection, which you then program (you need a SIO device) onto the cartridge.

Price: 76 Euros (including P&P) - about £68. Contact JAC! on the AtariAge forum to order. Also available at a cheaper price uncased and you can 3D print the shell yourself.

Ultimate Cart - a cartridge with an SD card slot that can emulate a wide range of cartridges (up to 1024k in size) from files on the SD card. It can also launch XEX files direct from the SD card.

Price: $95 (about £70) plus postage. Currently being produced by MacRorie in the USA (***thebrewingacademy.com***).

UnoCart - a little brother to the Ultimate Cart, this emulates cartridges up to 128k in size and uses a microSD card instead. It can also load XEX files and provides limited emulation of a floppy drive on 64k machines.

Price: $55 (about £40) plus postage. Currently being produced by MacRorie in the USA (thebrewingacademy.com). Or build it yourself – instructions are in issue #4 of Excel magazine!

SIDE2 - adds hard-disk functionality to your Atari using an inserted Compact Flash (CF) card. Can also launch XEX files directly from a DOS formatted partition of the CF card.

Note that the hard disk functionality requires an Atari with at least 128k for proper operation.

Price: 70 Euros (including P&P) – about £62, from Lotharek in Poland: ***lotharek.pl***

MyIDE II - hard disk emulation and more using a Compact Flash card. This cartridge can also load XEX and ATR files.

Price: $70 (about £52) plus postage from ***atarimax.com***

Memory Expansion

Ultimate 1MB 2014 - a 1 meg memory expansion for your Atari, real time clock, and also provides a PBI hard disk interface in conjunction with the SIDE2 cartridge.

Note this requires internal installation in your Atari and may require some soldering.

Price: 71 Euros including postage (about £63) from Lotharek in Poland - ***lotharek.pl***.

SysCheck v2.2 - 512k external RAM upgrade and OS switcher supporting 4 OS ROMs. One of the build in ROMs provides System checking diagnostics.

This can be used on a XE (with ECI slot) or XL computer.

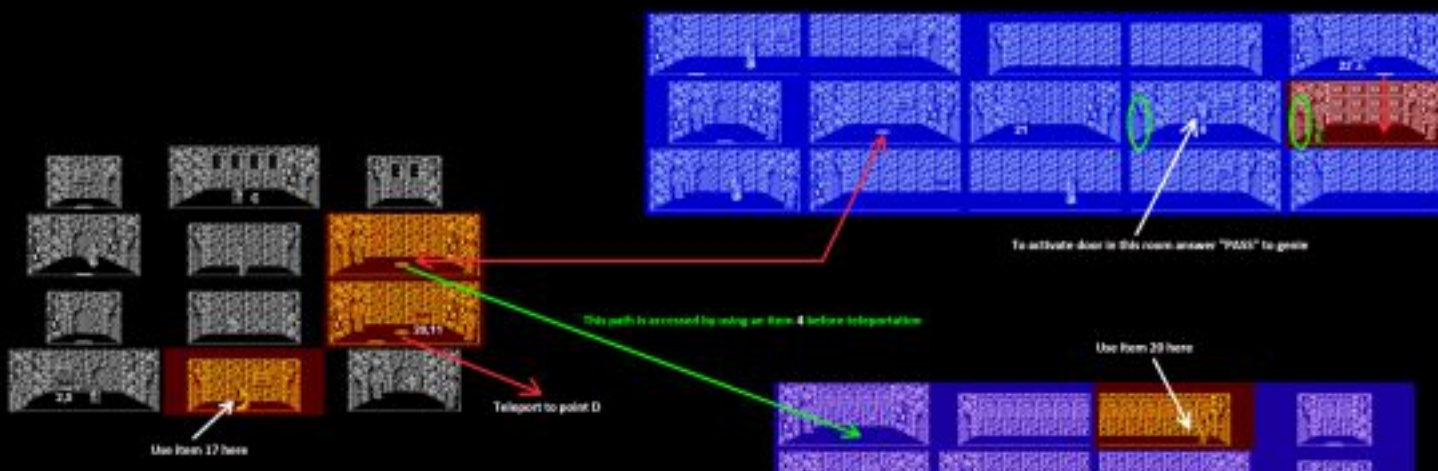
Price: 52 Euros including postage (about £46) from tf_hh on AtariAge forums.suppliers on ebay from about £40.

Disk Drive Replacements

SIO2SD - a modern replacement for a floppy drive. It can emulate 8 drives at once, reading from ATR files (disk images) on an SD card inserted into the device.

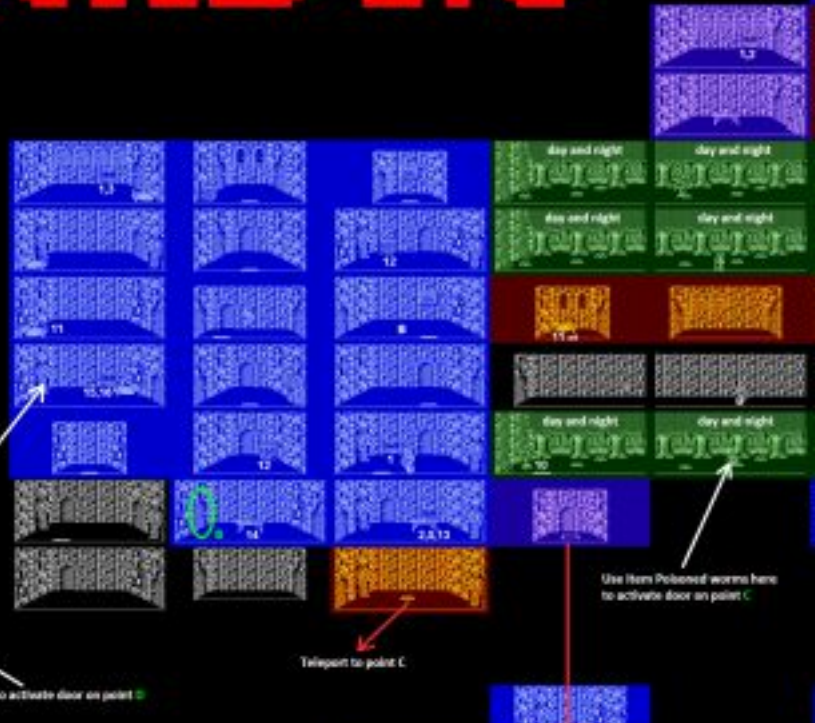
Price: the model pictured is from Lotharek in Poland, priced at 90 Euros (including P&P) – about £80. ***lotharek.pl***

Cheaper uncased models are available from other suppliers on ebay from about £40.



Spellbinder

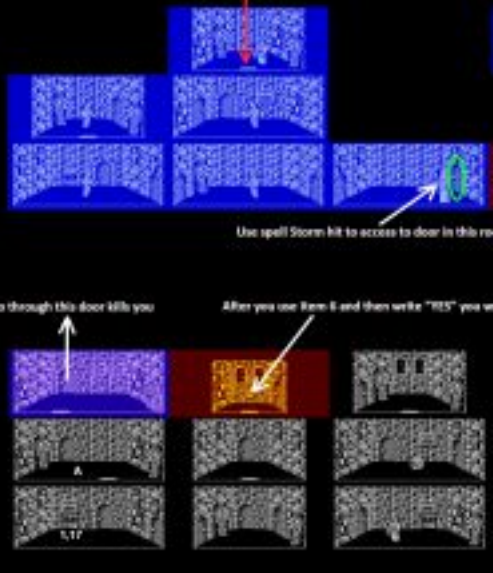
1 - Burned ashes
2 - Toad legs
3 - Dried roots
4 - Telechange
5 - Sulphur
6 - Golden bar
7 - Cork
8 - Air shield (spell)
9 - Blue mushroom
10 - Oak leaves
11 - Fleas
12 - Wine
13 - Mortar & Pestle
14 - Worms
15 - Heal (spell)
16 - Wind shield (spell)
17 - Lute
18 - Deep freeze (spell)
19 - Red herring
20 - Golden cross
21 - Crystal ball
22 - Golden fruit
23 - Tap
24 - Freeze (spell)
25 - Black pearl
26 - Golden crown

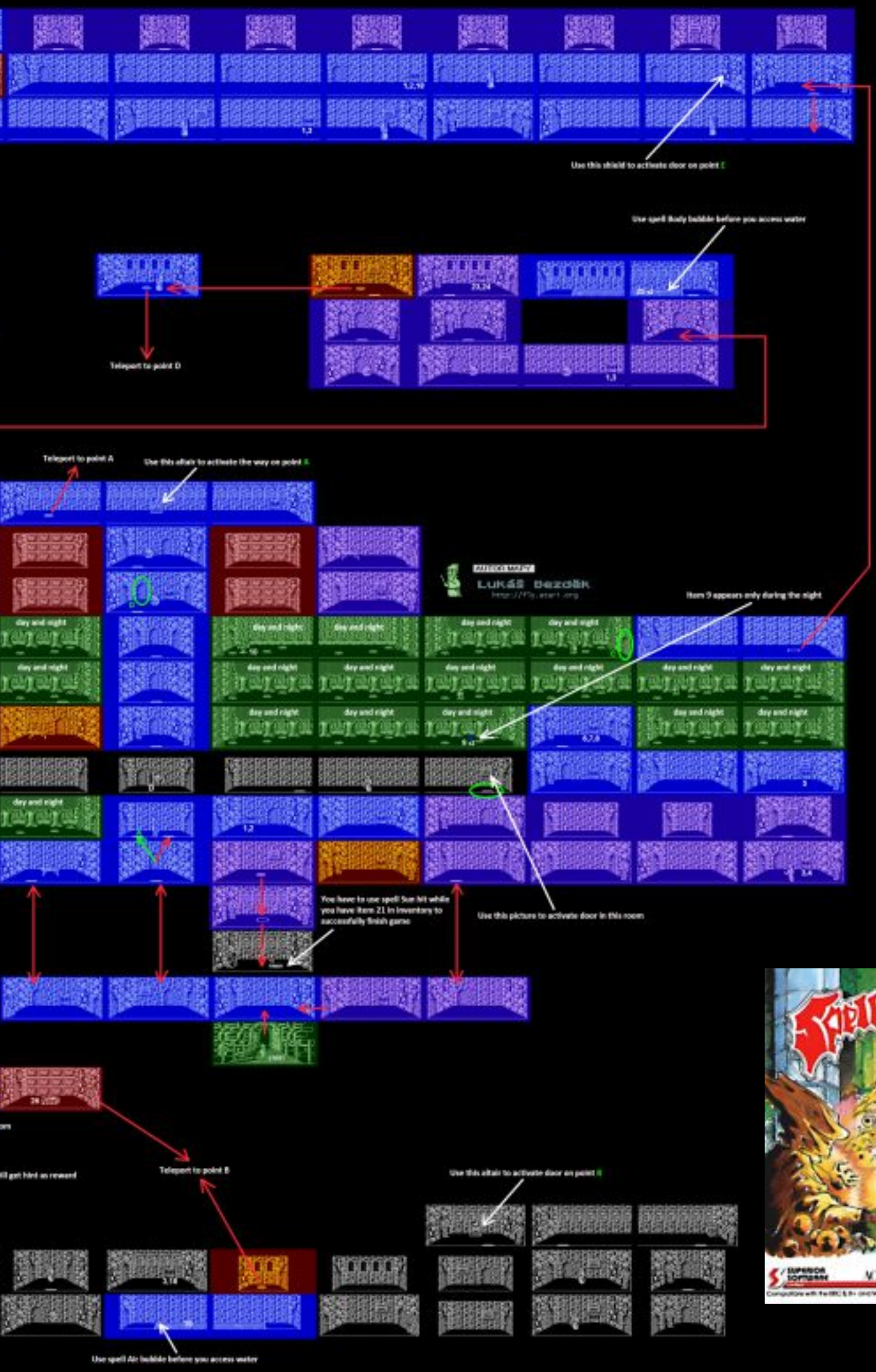


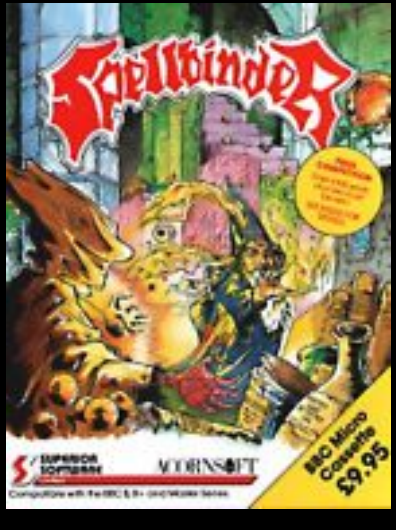
To make these recipes you need to have an item 13 in inventory

Recipes:

10 + 3	=	Air bubble
10 + 3 + 11	=	Body bubble
9 + 1 + 1 + 11	=	Storm hit
22 + 2 + 2 + 11	=	Thunder hit
25 + 1 + 11 + 11	=	Sun hit
14 + 5	=	Poisoned worms
5 + 3	=	Heal (partly refills energy)
5 + 3 + 11	=	Full heal (refills energy to 100%)
2 + 1	=	Freeze (slows down enemy movement)
2 + 1 + 11	=	Deep freeze (stops enemy movement)
12 + 1	=	Air shield (enemies take less energy when hit)
12 + 1 + 11	=	Wind shield (enemies take no energy when hit)




Use this shield to activate door on point C
Use spell Body bubble before you access water
Teleport to point D
Teleport to point A
Use this altair to activate the way on point A
Item 9 appears only during the night
You have to use spell Sun bit while you have item 21 in inventory to successfully finish game
Use this picture to activate door in this room
Teleport to point B
Use this altair to activate door on point B
Use spell Air bubble before you access water


Spellbinder
BBC Micro Cassette £9.95

Paul Lay Interview by Jonathan Halliday

Jon: Hi Paul, and thanks for agreeing to be interviewed. I wondered if we could perhaps work our way up to “Atari Blast” in the context of all your previous work. Perhaps you could begin by providing a precis of your life with special reference to the Atari, explaining how you first became interested in the A8, and how this relates (if at all) to your chosen profession.

Paul: My first experience of Atari, other than the arcades, was my uncle's VCS. I would have been about 12 years old and loved playing the games with my brother and uncle. Basketball, Miniature Golf and Superman were some of my favourites. I probably played games more back then than I have ever since so they remain my fondest gaming memories.

My uncle brought an 800 when they arrived in the UK, together with the BASIC and Assembler Editor carts and a copy of De Re Atari. They didn't get much of a look in at first with all our time spent playing Star Raiders.

I started doing Computer Science at school and it was a lot more interesting trying to program my uncle's 800 than it was the RM 380Z or Commodore PET we had at school. For me it was De Re Atari that made it so much fun, it was concise and easy to read, gave simple examples and suggested what could be done with more advanced techniques. Without the A8 and De Re Atari I don’t know if Computer Science at school would have been interesting enough for me to want to pursue a career in software.

I do remember taking David's Midnight Magic back to the shop with my uncle, complaining that it was in black and white but the booklet showed it in colour. I had not yet got to the section about NTSC artifacting.

One idea suggested in De Re Atari was to alternate display lists every VBI to superimpose one upon the other. I wrote a program to do that and sent it off to PCW magazine who published it. The payment from that made a good contribution towards getting my own 400.

I continued learning to program on my 400, sending programs off to PCW and Page 6 magazines, with the better ones getting published. One program in Page 6 was "Graphics Impossible" which did a mid-scanline GTIA mode/player HPOS change. In response to that I received a letter from Harvey Kong Tin from New Zealand, who was working on Hawkquest with his friend Andrew Bradfield. We did end up working together after he had finished Hawkquest but on other systems, we never did anything on the A8 together back in the day.

One summer holiday I decided to try writing something more than a magazine game and Sprong was the result. I took it around one of the London computer shows and ended up with it being published by Big Nose Software. Funnily enough the two guys from Bignose, Steve Calkin and Peter Sleeman, both lived locally to me. They also both worked for Ford Motor Company which is sort of where I work now (the company is called Visteon, but it was spun off from Ford in order to do work for other manufacturers).

Peter Sleeman ended up working for a software publisher and via him I was contacted by Jon Dean. I don't remember if he was still working for Atari UK or Activision at the time, but he offered me the chance to work on a Blue Peter game. They had run a competition to design a game, with the winner's design getting programmed and published. I don't remember much about it, I seem to recall a number of 3D rooms. I turned the opportunity down as I had started college and didn't want it to interfere with my studies.

I moved onto the ST when it came along. I didn't keep any of my old A8 stuff because I never thought I would ever return to it. At some point I bought a 130XE and a SIO2PC cable but that was just to transfer some files for Harvey. However, I did keep hold of it, storing it up in the loft.

It was Christmas 2011 and I was getting some decorations down from the loft. I saw the 130XE and wondered if anyone was still writing software for it. I googled, found Crownland and was blown away. The graphics were far beyond anything I'd ever seen on the A8 and I was fascinated by how the player missile graphics were used. I googled some more and discovered the concept of sprite multiplexing. That is what got me back to the A8, to see if I could write a sprite multiplexer.

I did end up with a working sprite multiplexer but then I lost interest. I picked it up again a year later and started using it in a little shooter game. I had not been in contact with Harvey for years and he just happened to email me at that time. I told him you won't believe it, but I've actually done some A8 coding - and that is how the two of us started working on what was to become AtariBlast!

Jon: You mention BASIC and Assembler Editor cartridges. Presumably – like most of us – you gradually transitioned from BASIC to assembly language, but did this involve a mix of languages for a while (BASIC with assembly language subroutines), or did you quickly jettison BASIC entirely? And since you were using different flavours of BASIC at school, did you find the limitations of Atari BASIC (lack of arrays, etc) especially restrictive?

Paul: Yes, I began with BASIC and started using assembly routines with it, my first few Page 6 programs are this format. I always found Atari BASIC a little quirky, but I don't think we had a really good BASIC at school until the BBC micros came along.

Jon: What about the other BASICs which came later, like Turbo BASIC XL, BASIC XE, etc? Or had you fully transitioned to assembly language by then? I took a circuitous route: Atari BASIC (with bits of assembly language), Turbo BASIC XL (with bits of assembler), then CC65 (the old native A8 version), AMAC, and finally an assembler I wrote based on Charles Marslett's A65.

Paul: No, I never used any other BASICs but I did switch from the Assembler Editor cart to AMAC.

Jon: I know a lot of MAC/65 users liked DDT, but when I used AMAC I didn't use a machine language monitor at all. How did you handle machine-level debugging in the old days?

Paul: I didn't use a monitor either. I would just test after every little change so that if I did make a mistake I'd

know where to look. I had to work like that on the SNES too, still do today really.

Jon: Did you find yourself having to write many development tools?

Paul: Most of the tools I wrote were for character sets and PM graphics. I know I wrote a GTIA 9 editor because Harvey used it for some graphics in HawkQuest (and gave me a credit for it) although I don't remember much about it now.

One thing I do remember is a BASIC program where I had each key play a note. I am not musical at all, but whenever I found a few key presses that sounded OK I would write them down. When I had enough such sequences I'd try and join them together. That is how I did my "music". I was so pleased (and I still am today, given how it was done) with the "music" on Heavy Metal.

Jon: I am musical but am all fingers and thumbs trying to do anything with sound on the Atari, so you're to be commended. I checked out the music at the end of "Heavy Metal" and it has an impressionistic, chromatic feel, which offers an interesting contrast with the mention of Jethro Tull's "Crest of a Knave". I take it this reflected a personal interest in progressive rock or folk?

Paul: Well if I could have made some music that resembled a Jethro Tull song I would have used it! Yes, I was into progressive rock and they were my favourite band.

Jon: You mention college: did you pursue IT studies there? If so, how relevant do you think your 8-bit endeavours were to development on other platforms (16-bit and beyond), and to what extent – if any – have you transferred those skills to modern software development?

Paul: Yes, I did Computer Science at college. I think knowing assembler is a useful skill even today. You may not actually develop in assembler but for those really difficult bugs you may well end up single stepping through assembler instructions. Also for any embedded work the basics learnt from the A8 still apply today.

Jon: Do you still develop in-situ on the Atari or do you find cross-development easier now (your high regard for Altirra suggests the latter)? Which is your preferred A8-targetted code editor/assembler today?

Paul: I cross-develop nowadays. I chose the ca65 assembler because the instructions were in English. I use Borland CodeWright for editing because I have used it for many years.

After the A8 I dabbled with the ST and PC for a while and then ended up doing some homebrew on the SNES. To me that felt like working on an A8 successor, you had the 65C816 together with these fantastic graphics and audio processors.

Jon: Upon returning to the scene, you were clearly confronted by many exciting developments which had taken place during your Atari hiatus. Aside from Crownland, what projects or technical accomplishments have impressed you most?

Paul: The thing which impressed me most was undoubtedly Altirra. It is incredible that someone had put so much effort into emulating the A8 so accurately (and providing an excellent debugging interface). On the hardware front, at first I thought it was great being able to read SD cards via SIO2SD although now I prefer the Ultimate Cart.

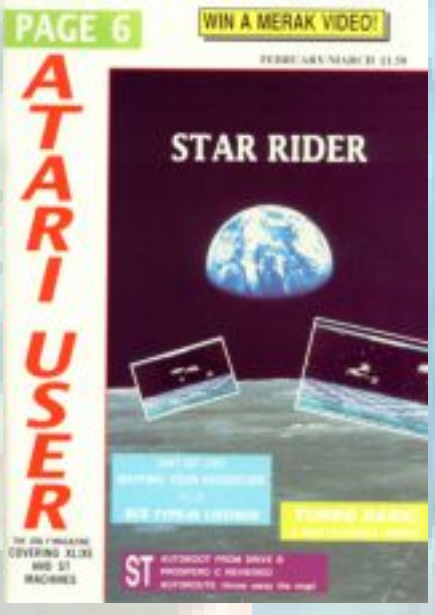
Jon: Do you think the 8-bit still holds many secrets, and - indeed - how long do you think the current rate of innovation can be sustained?

When I returned to the scene in late 2008, I felt lucky to have arrived at what seemed to be the beginning of an A8 Renaissance, but there have been signs of things slowing down. How healthy do you think the 8-bit Atari scene is today?

Paul: Currently it does seem like there is a lot more activity in hardware than there is in software, but I think that is OK since a large part of the scene is playing the old games you played as a kid. That aspect you imagine will last for a long time and I'm sure there will be a resurgence in software - when you release the GUI!

Jon: What's your feeling about VBXE, Rapidus (65C816 accelerator), etc? Do you think such upgrades fundamentally alter the character of the Atari and in so doing remove some of the interesting development challenges, or do you think developers should embrace enhanced hardware?

Paul: I imagine if I'd stuck with the A8 as my hobby machine for the last 35 years then I would be itching to do something with VBXE or Rapidus.



Although it was not the original intent I ended up targeting a 1MB cart or memory expansion for AtariBlast! It was interesting finding out about bank switching and the game certainly benefits from it. I think anyone writing software for the A8 in this day and age should do whatever they get the most enjoyment from. The user base is so small I don't see that it matters whether it is 25 or 250 people that can run your program, unless you are looking to sell it.

Jon: You clearly work well with Harvey, but I wonder about your working practices. Do you have any particular toolchain setup or online collaboration tools? When one person has time to code but the other is tied up with real life matters, how do you go about progressing things?

Paul: I think this was the first time working with Harvey that we had email and Internet. When we worked together before, even on the SNES projects, we sent disks to each other through the post (which took about 2 weeks between the UK and NZ).

For AtariBlast! I wrote a player editor and a level designer (both in ActionScript) which Harvey could run from a browser or run locally. The level designer could generate a playable Atari executable and there was also some "stitch" software that would take all the files and combine them to generate the full game. So Harvey was able to work on the graphics independently from me.

At one point I got completely fed up with AtariBlast! and didn't touch it for a year. During this time Harvey continued with the graphics design, because he was enjoying it I think. Without Harvey I would have abandoned it but I felt a responsibility to him because of all the work he had put into the graphics.

Jon: You mentioned the GUI project before and I can definitely identify with that sense of

responsibility to get things done, despite frequent and long hiatuses. Do you think sharing works in progress on forums can also help with motivation? And do you feel - when returning to a project which has been long neglected - that it takes time to get back into the right frame of mind?

Paul: I think sharing works in progress is a mixed bag. Positive responses are motivating and there can be some useful ideas but sometimes it feels like you are having to justify what you are doing with your own free time.

I'm not sure I ever got back into the right frame of mind with AtariBlast!, I just wanted to get it finished. Really, the last 6 months of working on it, was just grit my teeth and get it done.

Jon: How do your work colleagues react when (and if) they learn that you write games for an 8-bit computer?

Paul: I recently found out that one of my colleagues used to own an 800XL! He couldn't believe there was still interest in them today.

Jon: Finally, do you have your eyes set on any future projects for the Atari?

I have prototyped a few things on the A8 this year but none of them have jumped out as something to pursue. I would like to do something for the New Year Disk but time is running out.

Jon: Thank you so much for your time!

Next issue:
We'll have an interview with the mighty Tezz (Terrence Derby), who had a hand or two in the superb Atari productions of Boulderdash 25th Anniversary Edition, Saboteur, Chimera+, Bomb Jack, Jack the Nipper and the wonderful, wonderful Manic Miner! And soon... Barbarian!

Screenshots from Paul's 2015 New Year game "Strictly Gone Bananas"

Paul's Boulderdash clone "Rockfall" for the Super Nintendo

COMING SOON

8-BIT SLICKS

TIME PILOT

SPACE FORTRESS OMEGA

BARBARIAN

HELICOMMANDER